UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Dpto. de INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

INGENIERÍA TÉCNICA INDUSTRIAL
ESPECIALIDAD ELECTRÓNICA INDUSTRIAL

PROYECTO FIN DE CARRERA

MODELADO DEL ROBOT HUMANOIDE RH-2 EN LA PLATAFORMA DE SIMULACIÓN OPENHRP

AUTOR: RUBÉN MANUEL SIERRA MOLINA

TUTOR: CONCEPCIÓN ALICIA MONJE MICHARET

UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Dpto. de INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

INGENIERÍA TÉCNICA INDUSTRIAL
ESPECIALIDAD ELECTRÓNICA INDUSTRIAL

PROYECTO FIN DE CARRERA

MODELADO DEL ROBOT HUMANOIDE RH-2 EN LA PLATAFORMA DE SIMULACIÓN OPENHRP

AUTOR: RUBÉN MANUEL SIERRA MOLINA

TUTOR: CONCEPCIÓN ALICIA MONJE MICHARET

A mis abuelos
Gregoria, Carmen,
Ignacio y Delfín.
Os tenemos presentes.

Agradecimientos

Hace cinco años que comencé a estudiar la Ingeniería Técnica Industrial, que no comenzó nada bien para mí, por los malos resultados del primer año. Me costó mucho entrar en la dinámica, y tras mucho esfuerzo de por medio al fin me encuentro redactando los agradecimientos del proyecto fin de carrera.

No puedo evitar acordarme de mis amigos del instituto, Gonzalo, David y Juan, de mis amigos de la Universidad Carlos III, como Nacho y en especial mi amigo Carlos, el que ha sido uno de mis dos compañeros de proyecto y amigo desde el principio de la carrera, por todo lo que me has aguantado durante estos años y por las risas y los buenos momentos vividos, además de todas las personas con las que he pasado buenos momentos y espero seguir pasándolos.

A mi tutora Concha, ha sido un placer compartir estos meses de trabajo. Gracias por estar siempre cuando te hemos necesitado, y por tirar de nosotros hacia delante en ciertos momentos en los que era muy complicado compaginar todo a la vez, y parecía que el proyecto se quedaba siempre en un segundo plano. Por supuesto también agradecer a Santi y Ramón, por el esfuerzo que están realizando en este sprint final para terminar la carrera.

A mi familia, mi madre, mi hermana, mi padre y mi hermano, por poder recibir siempre vuestro apoyo cuando lo he necesitado, gracias por quererme y cuidarme tanto, os quiero mucho a todos. No puedo olvidarme tampoco de personas que han sido clave en mi forma de ser como son mi tía Felicia y mi tío Anastasio, Piedad, Susana y María José, sin sus consejos, no habría llegado tan lejos.

A mis compañeros de trabajo de SerTec, en especial a Marcos, Borja, Ramón y Sara, que gracias a ellos mi adaptación al mundo laboral ha sido mucho más fácil, además de saber provocarme una sonrisa cuando lo he estado pasando mal. Gracias por todo.

RUBÉN MANUEL SIERRA MOLINA

22 de Julio de 2010

INDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 Introducción

1.1 Introducción

Hoy en día la simulación de elementos mecánicos se hace imprescindible debido a la complejidad de los prototipos diseñados, así como del coste al que se arriesgan los creadores, por ello, poder comprobar en un ambiente de realidad virtual lo que sucedería con ciertos elementos, variándolos, y haciendo infinidad de pruebas, ahorra muchos recursos y cantidad de tiempo.

Un simulador será necesario para:

- Observar el movimiento de los mecanismos
- Reproducir cada una de las partes del sistema o proceso de la forma más realista posible, pero siempre manteniendo bajo control la complejidad del modelado.
- Conocer sus fuerzas internas
- Determinar sus pares, e infinidad de datos facilitados por sensores colocados en ubicaciones puntuales, los cuáles sirven para saber qué elementos debemos utilizar para nuestro modelo.

Actualmente existen herramientas de simulación virtual ajustadas al nivel del usuario, con una interfaz entendible y previsible, la cual facilita el trabajo con el software y el control por parte del usuario, y no es necesario conocer lenguajes de programación informáticos ni adquirir conocimientos avanzados.

1.2 Objetivos

Este proyecto tiene como objetivo el modelado de la plataforma robótica Rh-2 en el simulador OpenHRP3. Dicho modelado permitirá la simulación de tareas del robot Rh-2 en el entorno virtual, permitiendo comprobar las características necesarias para el movimiento del mismo y de esta manera, obtener los requisitos necesarios para su fabricación, tales como par mínimo que debe tener cada motor de la articulación, etc.

En primer lugar, se realizará un estudio de dicho simulador y de sus características principales. Es decir, los programas que necesita el simulador OpenHRP3 para ejecutarse la interfaz gráfica y estructura interna del simulador.

Posteriormente, se desarrollará el modelo VRML del robot Rh-2 para la plataforma OpenHRP3 siguiendo todas las características físicas, cinemáticas y dinámicas de dicho robot.

A continuación, una vez desarrollado el modelo se simulará un paso del robot Rh-2 en la plataforma OpenHRP3, determinando así las fuerzas a las que es sometido en las distintas partes del cuerpo en diferentes momentos críticos, la estabilidad que tiene ante las trayectorias programadas, etc.

1.3 Estructura del documento

El capítulo 1 "Introducción" comienza con una breve introducción al proyecto y nos sitúa en el entorno en el que se va a desarrollar. Además, se exponen los objetivos principales y la estructura del mismo.

En el capítulo 2 "Descripción del simulador OpenHRP3" se trata todo lo referente a la plataforma OpenHRP3. Se explican los programas que son necesarios para el funcionamiento del simulador así como la estructura interna del simulador (servidores) y su interfaz gráfica.

En el capítulo 3 "Modelo de la plataforma Rh-2 en OpenHRP3" se explican las características del robot Rh-2, así como el modelo desarrollado para la plataforma OpenHRP3 de este robot y el código VRML que es utilizado para realizar dicho modelo.

El capítulo 4 "Simulación de tareas en OpenHRP3" contiene la información de los archivos necesarios para la simulación de tareas en la plataforma OpenHRP3 y los pasos a seguir para realizarla.

El capítulo 5 "Simulación de tareas del robot RH-2 en OpenHRP3" explica dos tareas realizadas en el simulador OpenHRP3. Éstas consistirán en realizar un paso del robot a través de unas trayectorias dadas. Se comprobará las diferencias entre uno y otro y se mostrará la respuesta del robot.

Las conclusiones a las que se han llegado se muestran en el capítulo 6. Mientras que los posibles trabajos futuros y líneas de investigación que se pueden seguir se muestran en el capítulo 7.

Al final del documento, se incluyen la bibliografía utilizada y los anexos.

2 Simulador OpenHRP3

2.1 Descripción general

OpenHRP3 (Open Architecture Humanoid Robotics Plataform version 3) es una plataforma para simulaciones de robots y desarrollo de software. Permite a los usuarios inspeccionar el modelo original del robot y el programa de control a través de una simulación dinámica. Además, OpenHRP3 proporciona diversos componentes software de cálculo y bibliotecas que pueden ser utilizadas para desarrollar software relacionados con la robótica.

OpenHRP3 se desarrolla dentro de un importante programa de investigación impulsado por el Ministerio de Economía e Industria de Japón. Forma parte, como proyecto complementario, del llamado "Distributed component type robot simulator", llevado a cabo por "Cooperation of Next Generation Robots", que pertenece al Gobierno de Cooperación de Ciencia y Tecnología. La ingeniería de cálculos dinámicos es desarrollada por "Nakamura Lab, Dept. of Mechano Informatics, University of Tokyo" y la interfaz gráfica es realizada por "General Robotix, Inc". Las otras partes se desarrollan como trabajo entre cooperación de "Humanoid Resarch Group" y "Task-Intelligence Research Group" en los institutos "Intelligent Systems Research Institute" y "National Institute of Advanced Indrustrial Science and Technology (AIST)" [8].

Esta tercera versión (llamada OpenHRP3) ha mejorado considerablemente el desarrollo en comparación con la anterior versión OpenHRP2. Además, OpenHRP3 se distribuye como software de código abierto.

2.2 Requerimientos específicos

2.2.1 Programas utilizados

Para utilizar el simulador OpenHRP3 es necesario instalar unos programas específicos que serán descritos a continuación.

2.2.1.1 Visual Studio 2008

Microsoft Visual Studio 2008 [10] es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión net 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

El nuevo framework (.Net 3.5) está diseñado para aprovechar las ventajas que ofrece el nuevo sistema operativo "Windows Vista" a través de sus subsistemas "Windows Communication Foundation" (WCF) y "Windows Presentation Foundation" (WPF). El primero tiene como objetivo la construcción de aplicaciones orientadas a objetos, mientras que el último apunta a la creación de interfaces de usuario más dinámicas que las conocidas hasta el momento.

A las mejoras de desempeño, escalabilidad y seguridad con respecto a las anteriores versiones, se agregan entre otras, las siguientes novedades:

- La mejora en las capacidades de Pruebas Unitarias permiten ejecutarlas más rápido independientemente de si lo hacen en el entorno IDE o desde la línea de comandos. Se incluye además un nuevo soporte para diagnosticar y optimizar el sistema a través de las herramientas de pruebas de Visual Studio. Con ellas se podrán ejecutar perfiles durante las pruebas para que ejecuten cargas, prueben procedimientos contra un sistema y registren su

comportamiento, además de utilizar herramientas integradas para depurar y optimizar.

• Con “Visual Studio Tools for Office” (VSTO) integrado con Visual Studio 2008 es posible desarrollar rápidamente aplicaciones de alta calidad basadas en la interfaz de usuario (UI) de Office que personalicen la experiencia del usuario y mejoren su productividad en el uso de Word, Excel, PowerPoint, Outlook, Visio, InfoPath y Project.

• Visual Studio 2008 permite incorporar características del nuevo “Windows Presentation Foundation” sin dificultad tanto en los formularios de Windows existentes como en los nuevos. Ahora es posible actualizar el estilo visual de las aplicaciones al de Windows Vista debido a las mejoras en “Microsoft Foundation Class Library (MFC)” y Visual C++. Visual Studio 2008 permite mejorar la interoperabilidad entre código nativo y código manejado por .NET. Esta integración más profunda simplificará el trabajo de diseño y codificación.

• LINQ (Language Integrated Query) es un nuevo conjunto de herramientas diseñado para reducir la complejidad del acceso a Base de Datos, a través de extensiones para C++ y Visual Basic así como para Microsoft .NET Framework.

• Visual Studio 2008 ahora permite la creación de soluciones multiplataforma adaptadas para funcionar con las diferentes versiones de .Net Framework: 2.0. (Incluido con Visual Studio 2005), 3.0 (incluido en Windows Vista) y 3.5 (incluido con Visual Studio 2008).

• .NET 3.5 incluye biblioteca ASP.NET AJAX para desarrollar aplicaciones web más eficientes, interactivas y altamente personalizadas que funcionen para todos los navegadores más populares y utilicen las últimas tecnologías y herramientas Web, incluyendo Silverlight y Popfly.

En este caso, Visual Studio es el programa encargado de compilar todas las librerías necesarias que están incluidas en el proyecto OpenHRP3.

Algunas de las librerías que posee este proyecto son *Boost*, *Clapack* y *Tvmet*. A continuación se expone una breve explicación de cada una de ellas.

2.2.1.1.1 ***Librería BOOST***

BOOST [11] es un conjunto de librerías de código abierto preparadas para extender las capacidades del lenguaje de programación C++. Su licencia permite que sea utilizada en cualquier tipo de proyectos, ya sean comerciales o no.

Su diseño e implementación permiten que sea utilizada en un amplio espectro de aplicaciones y plataformas. Abarca desde librerías de propósito general hasta abstracciones del sistema operativo.

Con el objetivo de alcanzar el mayor rendimiento y flexibilidad se hace un uso intensivo de plantillas. BOOST ha representado una fuente de trabajo e investigación en programación genérica y meta programación en C++.

2.2.1.1.2 ***Librería CLAPACK***

En primer lugar se describe *Lapack* debido a que esta librería es la madre de CLAPACK [12]. *Lapack* es una librería de subrutinas escritas en Fortran77 para resolver la mayoría de los problemas que ocurren en álgebra lineal. Ha sido diseñada para ser eficiente en un amplio rango de computadores modernos de alto desempeño. El nombre *Lapack* es el acrónimo de Linear Algebra PACKage.

Lapack provee rutinas para resolver sistemas de ecuaciones lineales, problemas de mínimos cuadrados lineales, problemas de valor propio y problemas de valores singulares.

Por otra parte, funcionalidades similares son provistas para matrices reales y complejas, tanto en precisión simple como doble.

La librería CLAPACK fue construida utilizando la herramienta f2c que convierte código escrito en Fortran a código C. Para crear CLAPACK, fue necesario correr la librería completa de *Lapack* escrita en Fortran77 a través de f2c para obtener código C, y entonces ésta se modificó para mejorar la facilidad de su lectura.

El objetivo principal de CLAPACK consiste en proveer *Lapack* a usuarios que no tienen acceso a compiladores Fortran. Además, gracias a que CLAPACK tiene versiones tanto para LINUX como para Windows, puede ser utilizado por una gran cantidad de usuarios.

2.2.1.1.3 ***Librería TVMET***

TVMET [13] es una librería de vectores y matrices que usan Meta Templates y Expression Templates para evaluar los resultados en el tiempo de compilación.

El código producido es similar al código hand-code, pero la calidad del código todavía depende del compilador y de su versión. Las dimensiones para los vectores y las matrices son estáticas y limitadas.

2.2.1.2 Entorno OpenRTM

El OpenRT [9] trabaja la interfaz gráfica interactiva de 3D. Se encarga de los modelos dinámicos animados y de la iluminación global, para la visualización de un prototipo de alta calidad para juegos de ordenador.

El RT-MIDDLEWARE proporciona una plataforma común para la tecnología robótica y aumenta la eficacia de la investigación y desarrollo en la robótica y sus aplicaciones.

Un problema serio a menudo indicado en el desarrollo de software que apoya a los sistemas robóticos comparado con el software tradicional es la imposibilidad de reutilización debido a lo específico de cada sistema robótico. De esta observación, surge la idea de desarrollar un middleware que proporciona una plataforma común para ayudar en el desarrollo de sistemas robóticos, en los cuales un número grande de usuarios podría estar implicado, promoviendo así la reutilización del software de alto nivel en la realización de la tecnología robótica.
El proyecto de investigación y desarrollo de middleware causó varios datos específicos en el nivel de interfaz y la entrega de una puesta en práctica de un prototipo llamado OpenRTM-aist.

Para crear el entorno del proyecto, necesitamos el programa OpenRTM-aist (Advanced Industrial Science and Technology), el cual se ayuda de otros tres programas (ACE, OmniORB y Python). A continuación se hará una descripción de cada uno de ellos.

2.2.1.2.1 *ACE (Adaptative Communication Environment)*

El Ambiente de Comunicación Adaptable (ACE) [14] simplifica varios aspectos de la programación en red. Ofrece un juego de objetos de alto rendimiento orientados a clases de C++ diseñadas para dirigir las complejidades inherentes y desafíos de la programación en red, previniendo errores comunes.

ACE provee un camino estandarizado para el sistema operativo y trabaja con rasgos específicos para ser utilizados. Esto proporciona tipos de datos comunes y métodos para tener acceso a los rasgos poderosos pero complejos de sistemas operativos modernos. Estos incluyen: intercomunicación de los procesos, dirección de hilo, dirección de memoria eficiente, etc.

Fue diseñado para ser portable y proporciona un marco común. El mismo código trabajará sobre la mayor parte de Unix, Microsoft Windows, VxWorks, QNX, OpenVMS etc., con cambios mínimos. Debido a este portabilidad entre plataformas, ACE ha sido usado en el desarrollo de software de comunicación.

El Ambiente de Comunicación Adaptable (ACE) es un marco de código abierto, una herramienta orientada a objetos (OO) que pone en práctica un modelo principal para el software de comunicación simultáneo (concurrente). Las tareas de software de comunicación proporcionadas por ACE incluyen el acontecimiento “demultiplexing” y el envío de eventos; manejan la señal; atienden la inicialización; e interactúan con la comunicación, la dirección de memoria compartida, el envío de mensaje, la configuración dinámica de servicios distribuidos, la ejecución concurrente y la sincronización. ACE está destinado a los servicios de comunicación de alto rendimiento en tiempo real y sus diferentes usos. Esto simplifica el desarrollo de servicios que utilizan la comunicación de interproceso, el acontecimiento demultiplexing, la unión explícita dinámica, y la coincidencia. Además, ACE automatiza la configuración de sistema y la nueva configuración dinámicamente, uniendo servicios y ejecutando estos servicios en uno o varios procesos o hilos.

2.2.1.2.2 *OmniOrb (Object Request Broker)*

ORB es, en computación distribuida, el nombre que recibe una capa de software (también llamada middleware) que permite a los objetos realizar llamadas a métodos situados en máquinas remotas, a través de una red. Maneja la transferencia de estructuras de datos de manera que sean compatibles entre los dos objetos. Para ello utiliza un estándar para convertir las estructuras de datos en un flujo de bytes, conservando el orden de los bytes entre distintas arquitecturas. Este proceso se denomina marshalling (y también su opuesto, unmarshalling) [15].

Básicamente permite a objetos distribuidos interactuar entre sí de manera transparente, es decir, como si estuviesen en la misma máquina.

OmniORB es un ORB de alto rendimiento y robusto de CORBA para C ++ y Python. Está disponible libremente según los términos de licencia GNU.
En la mayoría de canales de comunicación hay una llamada “inflight” entre dos espacios de dirección en un momento dado. Para hacer esto sin limitar el nivel de coincidencia, se crean nuevos canales que conectan los dos espacios de dirección en demanda cuando hay llamadas concurrentes en curso. Cada canal es servido por un hilo dedicado. Este array proporciona la coincidencia máxima y elimina cualquier hilo que se pone en marcha en cualquiera de los espacios de dirección para tratar una llamada. Además, para maximizar el rendimiento cuando se trata de llamadas grandes, se envían elementos de grandes datos cuando son procesadas mientras otros están siendo ordenados.
De la versión 4.0 en adelante, omniORB también crea una unión de hilo flexible y hace posible varias llamadas en una sola conexión. Esto conlleva una pequeña cantidad de llamadas adicionales, comparado con el hilo por defecto del modelo de conexión, pero permite a omniORB escalar a números sumamente grandes de clientes simultáneos (concurrentes).

2.2.1.2.3 *Python*

Python [16] es un lenguaje de programación dinámico orientado a objetos que puede ser usado para muchas clases de desarrollo de software. Ofrece un gran apoyo a la integración con otros lenguajes e instrumentos y viene con bibliotecas estándar extensas.

Permite dividir el programa en módulos reutilizables desde otros programas Python. Viene con una gran colección de módulos estándar que se pueden utilizar como base de los programas (o como ejemplos para empezar a aprender Python). También hay módulos incluidos que proporcionan E/S de ficheros, llamadas al sistema, sockets y hasta interfaces a GUI (interfaz gráfica con el usuario) como Tk, GTK y Qt entre otros.

Los sistemas operativos sobre los que trabaja son Windows, Linux/Unix, Mac OS X, OS/2 y teléfonos móviles Nokia. Python también ha sido puesto a Java y máquinas .NET virtuales. Es distribuido bajo una licencia gratuita para utilizarlo libremente, aún para productos comerciales.

2.2.1.3 Java y Jython

2.2.1.3.1 *Java*

Java [17] es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un *bytecode*, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el *bytecode* es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del *bytecode* por un procesador Java también es posible.

Java es una aplicación que nos permite jugar en línea, participar en sesiones de chat con internautas de todo el mundo, calcular los intereses de una hipoteca y ver imágenes en tres dimensiones, entre otras muchas aplicaciones. Es también esencial

para las aplicaciones de intranet y otras soluciones de comercio electrónico que constituyen la base informática de las empresas.

Hasta la fecha, la plataforma Java ha atraído a más de cinco millones de desarrolladores de software. Se utiliza en los principales sectores de la industria de todo el mundo y está presente en un gran número de dispositivos, equipos y redes.

La versatilidad y eficiencia de la tecnología Java, la portabilidad de su plataforma y la seguridad que aporta la han convertido en la tecnología ideal para su aplicación a redes. De portátiles a centros de datos, de consolas de juegos a super equipos científicos, de teléfonos móviles a Internet, Java está en todas partes.

Java se ha convertido en un recurso inestimable ya que permite a los desarrolladores:

- Desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra plataforma
- Crear programas para que funcionen en un navegador web y en servicios web
- Desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, procesamiento de formularios HTML, etc.
- Combinar aplicaciones o servicios que usan el lenguaje Java para crear servicios o aplicaciones totalmente personalizados
- Desarrollar potentes y eficientes aplicaciones para teléfonos móviles, procesadores remotos, productos de consumo de bajo coste y prácticamente cualquier tipo de dispositivo digital

En OpenHRP3, algunos de los componentes, tales como GrxUI, están escritos en Java. Por lo tanto, tenemos que configurar el entorno Java para compilar y ejecutar los componentes.

2.2.1.3.2 *Jython*

Jython [18] es una aplicación de alto nivel, orientado a objetos en lenguaje Python escrito en Java. El predecesor de Jython, JPython, está certificado como 100% puro Java. Jython está disponible gratuitamente tanto para aplicaciones comerciales como no comerciales y se distribuye con su código fuente. Jython es complementario a Java y es especialmente adecuado para las siguientes tareas:

- Escritura integrada – Los programadores de Java pueden añadir Jython a bibliotecas de su sistema para permitir a los usuarios finales escribir simples o complicados scripts con el objetivo de añadir la funcionalidad de la aplicación.
- Interactivo con la experimentación - Jython proporciona un intérprete interactivo que puede utilizarse para interactuar con Java, con paquetes, o ejecutar aplicaciones Java. Esto permite a los programadores experimentar y depurar cualquier sistema usando Java Jython.
- El rápido desarrollo de aplicaciones – Los programas Python son típicamente entre 2-10X más corto que el equivalente programa Java. Esto se traduce directamente a un aumento de la productividad del programador. La interacción sin fisuras entre Python y Java permite a los desarrolladores libremente mezclar los dos idiomas.

2.2.2 Servidores

OpenHRP3 está formado por distintos servidores encargados de obtener los datos del modelo del robot y del entorno.

2.2.2.1 DynamicsSimulator

El simulador dinámico (DynamicsSimulator [8]) calcula la dinámica inversa del link, integrando la aceleración obtenida del joint (articulación) y actualiza la velocidad y valor del joint. Puede manejar arbitrariamente mecanismos en cadena abierta y mecanismos en cadena cerrada. Además, cuando detecta una colisión entre links, y si hay velocidad relativa entre ambos, en primer lugar ajusta la velocidad relativa a cero a través del cálculo de colisión y después calcula la potencia de contacto necesaria para no desarrollar una aceleración relativa. Una parte del resultado del cálculo dinámico puede ser tomado como salida del sensor.

2.2.2.2 CollisionDetector

El servidor CollisionDetector [8] detecta el contacto entre el robot y el entorno. Se llama desde el servidor DynamicsSimulator durante la ejecución de la simulación. OPCODE es utilizado como un algoritmo de detección de colisión. El usuario del servidor DynamicsSimulator no necesita llamar al servidor CollisionDetector directamente, sino que es llamado desde el propio DynamicSimulator si es necesario. El CollisionDetector calcula el punto de contacto y el vector de la fuerza de reacción.

2.2.2.3 ModelLoader

El ModelLoader [8] lee el modelo de robot con el que trabajará (descrito más adelante) el simulador y extrae los parámetros dinámicos y los datos de dicho robot. El modelo expresa los parámetros dinámicos y la dinámica del robot en el entorno del simulador mediante el uso de archivos VRML.

2.2.2.4 VisionSimulator

El servidor VisionSimulator [8] simula el sensor de visión (cámara) que adjunta el robot.

2.2.2.5 Controller

Controller [8] es un programa de control que incrementa el lazo de control del robot. Dicho lazo de control garantiza la estabilidad del prototipo durante la realización de tareas y funciona aún cuando el robot está parado.

2.2.2.6 CORBA

CORBA [8] (Common Object Request Broker Architecture) es un estándar definido por el Grupo de Dirección de Objeto (OMG) que permite componentes de software escritos en múltiples lenguajes de programación y controla múltiples ordenadores trabajando juntos. Es un mecanismo de software para normalizar la semántica de llamada de método entre los objetos de aplicación que residen en el mismo espacio de dirección (de aplicación) o remoto (el mismo anfitrión, o el anfitrión remoto sobre una red).

OpenHRP3 tiene un sistema de distribución de objetos basado en CORBA. Por este motivo todos los servidores en OpenHRP3 están implementados como objetos CORBA. Cada uno de los servidores están programados en el lenguaje y sistema operativo apropiado y finalmente conectados a través de CORBA.

2.2.3 Sistema Operativo

Actualmente OpenHRP3 es soportado en las siguientes plataformas [8].

- Ubuntu Linux 7 o posterior (recomendado).
- Windows XP, Vista (32bit).

2.3 Interfaz

En la
Figura 2.1 se muestra la interfaz del simulador OpenHRP3.

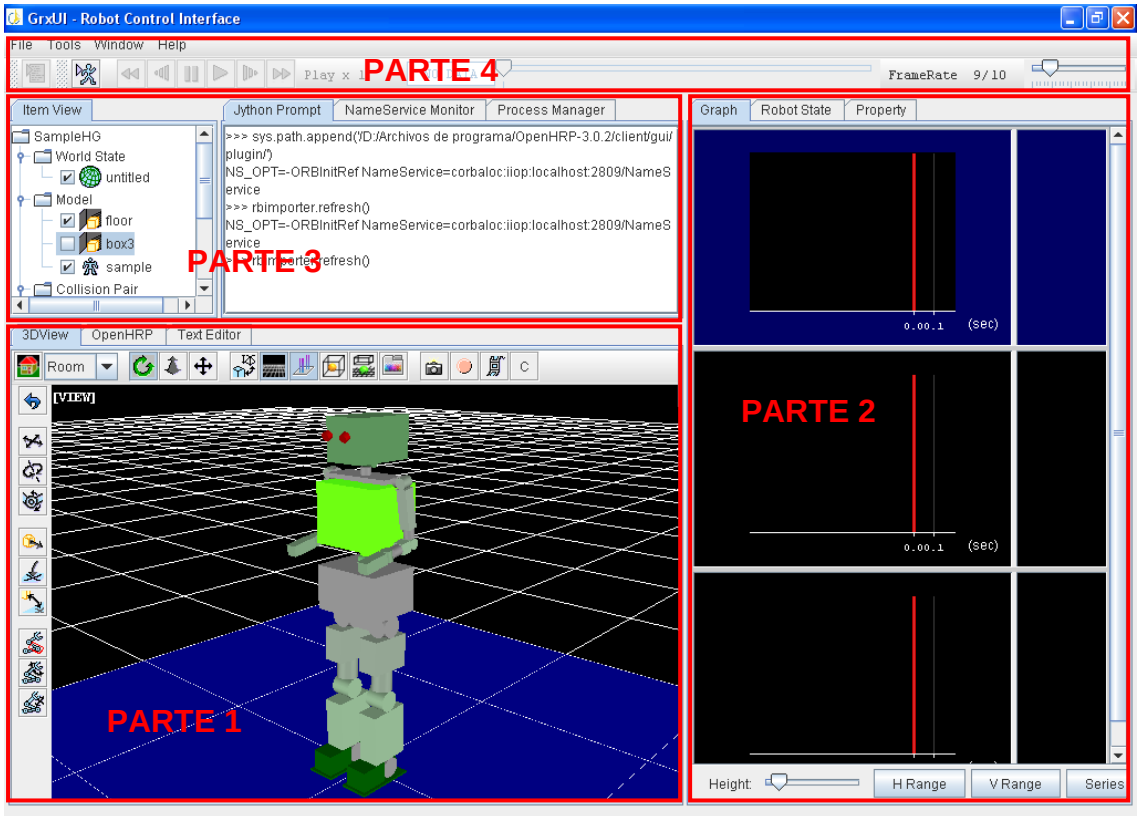


Figura 2.1 Interfaz simulador OpenHRP3

Podemos dividir la interfaz gráfica en cuatro partes. La primera de ellas está dirigida a la visualización del modelo del robot durante la simulación (Figura 2.2).

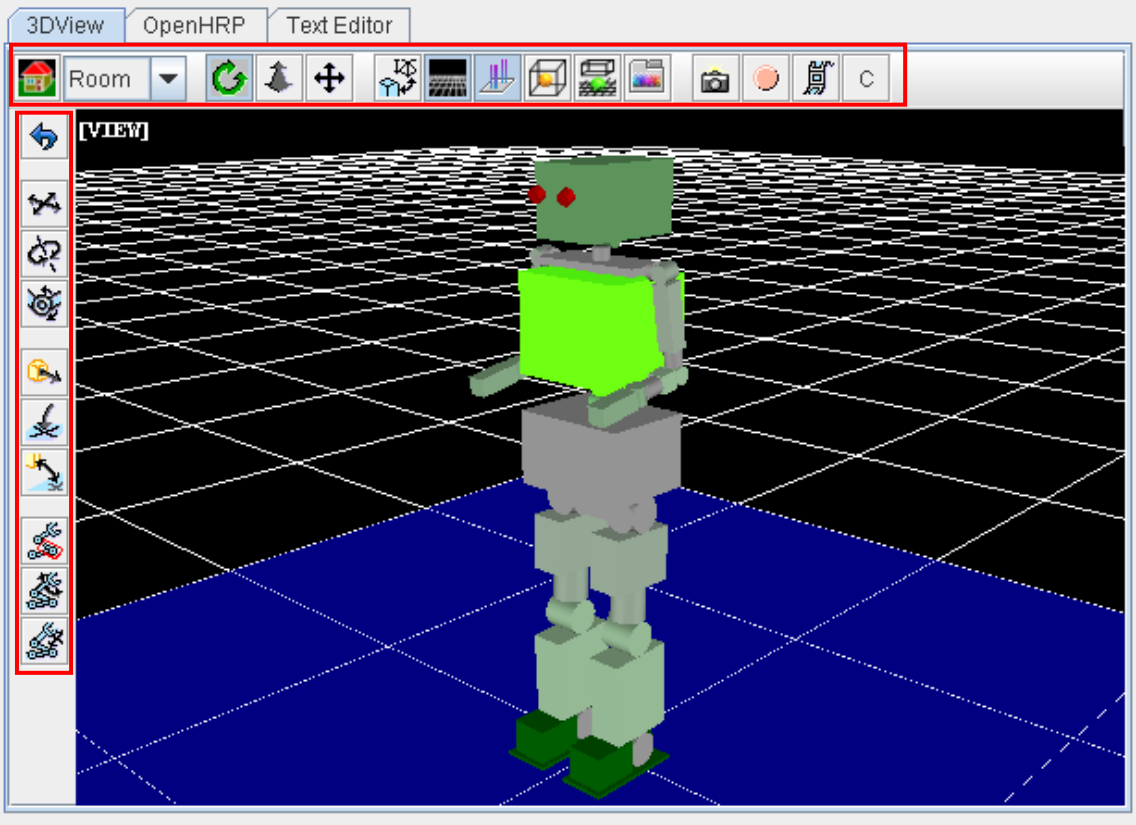


Figura 2.2 Visualización modelo

En esta zona se encuentran dos barras de herramientas, una vertical y otra horizontal. Las herramientas dispuestas horizontalmente nos permiten cambiar la vista del robot durante la simulación mediante los 3 ejes de coordenadas, ocultar planos de las figuras, ver los centros de gravedad del robot o de los objetos introducidos, además de tomar fotografías y videos de la simulación. Las verticales sin embargo, están orientadas al control de movimiento de una simple articulación, rotación de objetos determinados, comprobación de cinemáticas inversas de las figuras, etc.

La parte número 2, situada a la derecha de la interfaz, se visualizan las gráficas (Figura 2.3) que representan la evolución de las variables articulares durante la simulación (posición, velocidad, aceleración), así como los datos (Figura 2.3) de los distintos sensores y articulaciones incluidos a lo largo de la estructura mecánica del robot (sensores de fuerza, etc.).

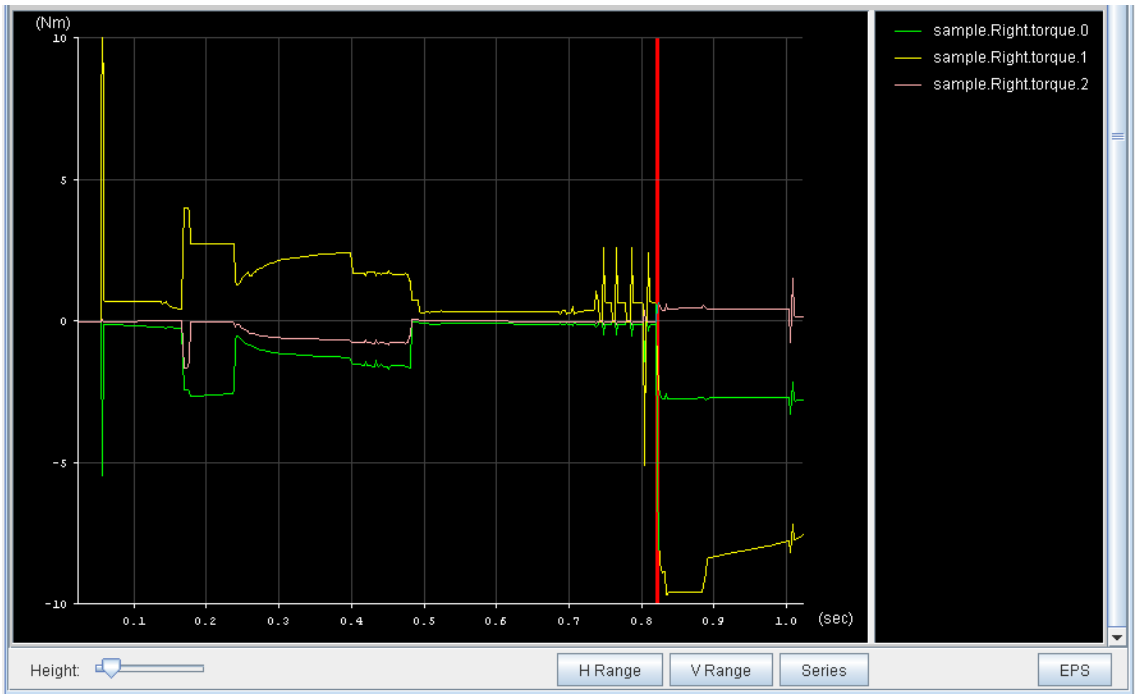


Figura 2.3 Vista de las gráficas

Graph | Robot State | Property

No	Joint	Angle	Target	Current	PWR	SRV	Pgain	Dg
0	RLEG_HIP_R	0.0	---	---	---	---	---	--
1	RLEG_HIP_P	-2.1	---	---	---	---	---	--
2	RLEG_HIP_Y	0.0	---	---	---	---	---	--
3	RLEG_KNEE	4.5	---	---	---	---	---	--
4	RLEG_ANKLE_P	-2.4	---	---	---	---	---	--
5	RLEG_ANKLE_R	0.0	---	---	---	---	---	--
6	RARM_SHOULD...	10.0	---	---	---	---	---	--
7	RARM_SHOULD...	-0.2	---	---	---	---	---	--
8	RARM_SHOULD...	0.0	---	---	---	---	---	--
9	RARM_ELBOW	-90.0	---	---	---	---	---	--
10	RARM_WRIST_Y	0.0	---	---	---	---	---	--
11	RARM_WRIST_R	0.0	---	---	---	---	---	--
12	LLEG_HIP_R	0.0	---	---	---	---	---	--
13	LLEG_HIP_P	-2.1	---	---	---	---	---	--
14	LLEG_HIP_Y	0.0	---	---	---	---	---	--
15	LLEG_KNEE	4.5	---	---	---	---	---	--
16	LLEG_ANKLE_P	-2.4	---	---	---	---	---	--
17	LLEG_ANKLE_R	0.0	---	---	---	---	---	--
18	LARM_SHOULD...	10.0	---	---	---	---	---	--
19	LARM_SHOULD...	-0.2	---	---	---	---	---	--
20	LARM_SHOULD...	0.0	---	---	---	---	---	--
21	LARM_ELBOW	-90.0	---	---	---	---	---	--
22	LARM_WRIST_Y	0.0	---	---	---	---	---	--
23	LARM_WRIST_R	0.0	---	---	---	---	---	--
24	WAIST_P	0.0	---	---	---	---	---	--
25	WAIST_Y	0.0	---	---	---	---	---	--
26	CHEST_Y	0.0	---	---	---	---	---	--
27	CHEST_P	0.0	---	---	---	---	---	--

Figura 2.4 Vista de los datos de articulaciones

En la parte número 3 de la interfaz se puede observar el árbol de módulos cargados en la simulación, tal y como se muestra en la parte izquierda de la Figura 2.5.

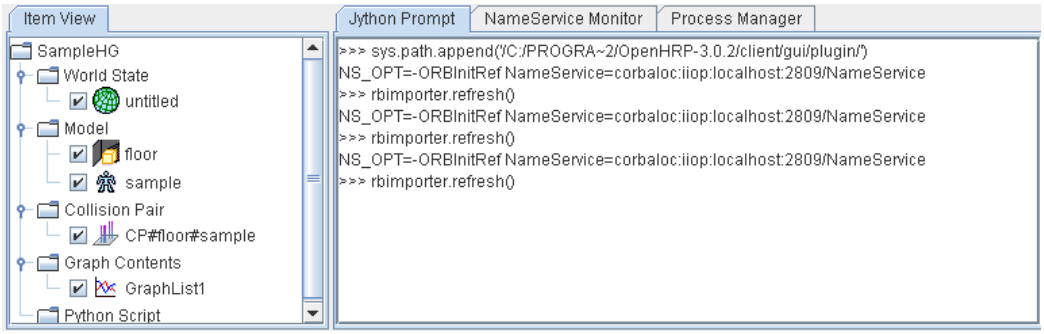


Figura 2.5 Árbol de módulos

Estos módulos pueden ser el modelo del robot, el modelo del entorno, el módulo de colisión (Collision Pair) y el paquete de visualización de gráficas, entre otros. En la derecha del árbol de módulos se encuentra la ventana “Jython Prompt”, donde se pueden introducir líneas de comando; la ventana “Process Manager”, donde podemos ver todos los procesos cargados por la interfaz de simulación; o la ventana “NameService Monitor” donde se observan los registros del servidor CORBA.

Por último se tiene la parte número 4, donde se puede manejar el inicio de la simulación, la parada de la misma, una vez obtenida se puede ver el resultado de la misma a una velocidad mayor; podemos cambiar el “frame rate”, el número de frames que se quiere que contenga el video de la simulación (en caso de querer grabarlo); además del menú donde podemos guardar el proyecto actual, crear uno nuevo, guardarlo, etc.

Figura 2.6 Barra de menú y controles de simulación

2.4 Arquitectura del simulador

Para entender mejor el funcionamiento del simulador y la relación que existe entre los diferentes archivos, etc, se adjunta un diagrama de bloques en el que se explica cual es el proceso de ejecución de un proyecto determinado, cabe destacar que muchas de las funcionalidades mencionadas en el diagrama serán explicadas en el Capítulo 4: Simulación de tareas en OpenHRP3:

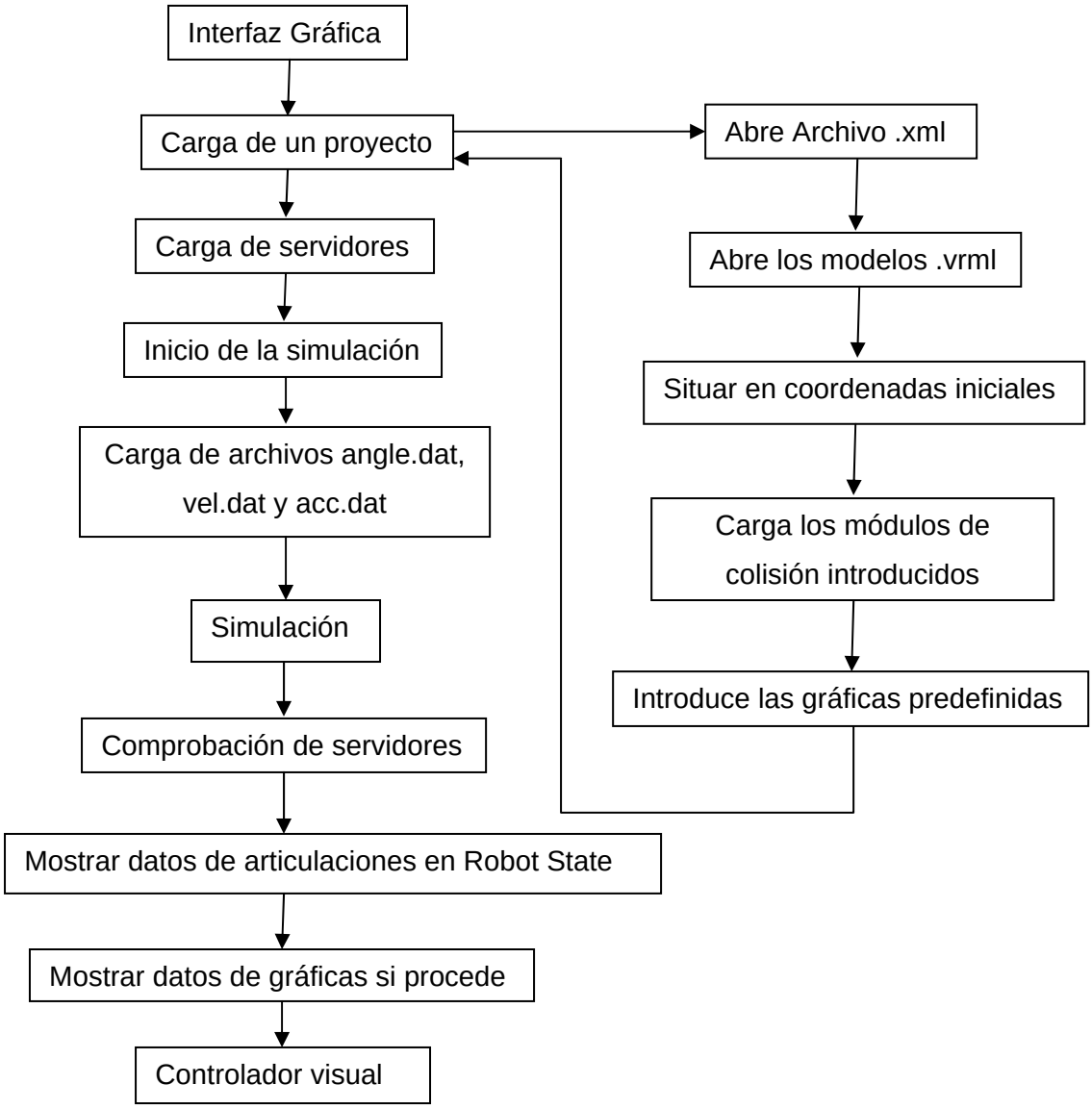


Figura 2.7 Arquitectura del Simulador

3 Modelo de la plataforma Rh-2 en OpenHRP3

3.1 Introducción al robot Rh-2

En el año 2004 nace el Robot Humanoide 0, el cual es el predecesor del actual robot humanoide de la Universidad Carlos III de Madrid, el robot Rh-1 [7]. Tras dos años de investigación, en 2006 el proyecto Rh-0 se transformó en el prototipo Rh-1. A día de hoy se trabaja en las mejoras de dicho robot para diseñar el Rh-2, un robot del que se espera que pueda llegar a subir y bajar escalones, transportar objetos, etc.

El robot Rh-1 (Figura 3.2) mide 1.52 m de altura, pesa alrededor de 50 Kg, y posee 21 grados de libertad. Su principal problema es que en cada paso debe soportar una fuerza de impacto de entre 700 y 800 Newton por segundo, lo que deteriora de manera importante con el paso del tiempo las articulaciones de las piernas del mismo.

Figura 3.1 Imagen del prototipo Rh-1

Por este motivo, y por las mejoras descritas anteriormente como que el robot sea capaz de subir y bajar escaleras surge la idea de la fabricación del prototipo Rh-2. Se pretende además que el Rh-2 camine de manera natural, pueda oír, imitar gestos y aprender habilidades, aunque este proyecto se centrará en el diseño del modelo para posteriormente simular el movimiento de andar.

El robot Rh-2 tendrá una altura de aproximadamente 1.50 m. Posee 28 grados de libertad (GDL) distribuidos como se muestra en la Figura 3.2 .

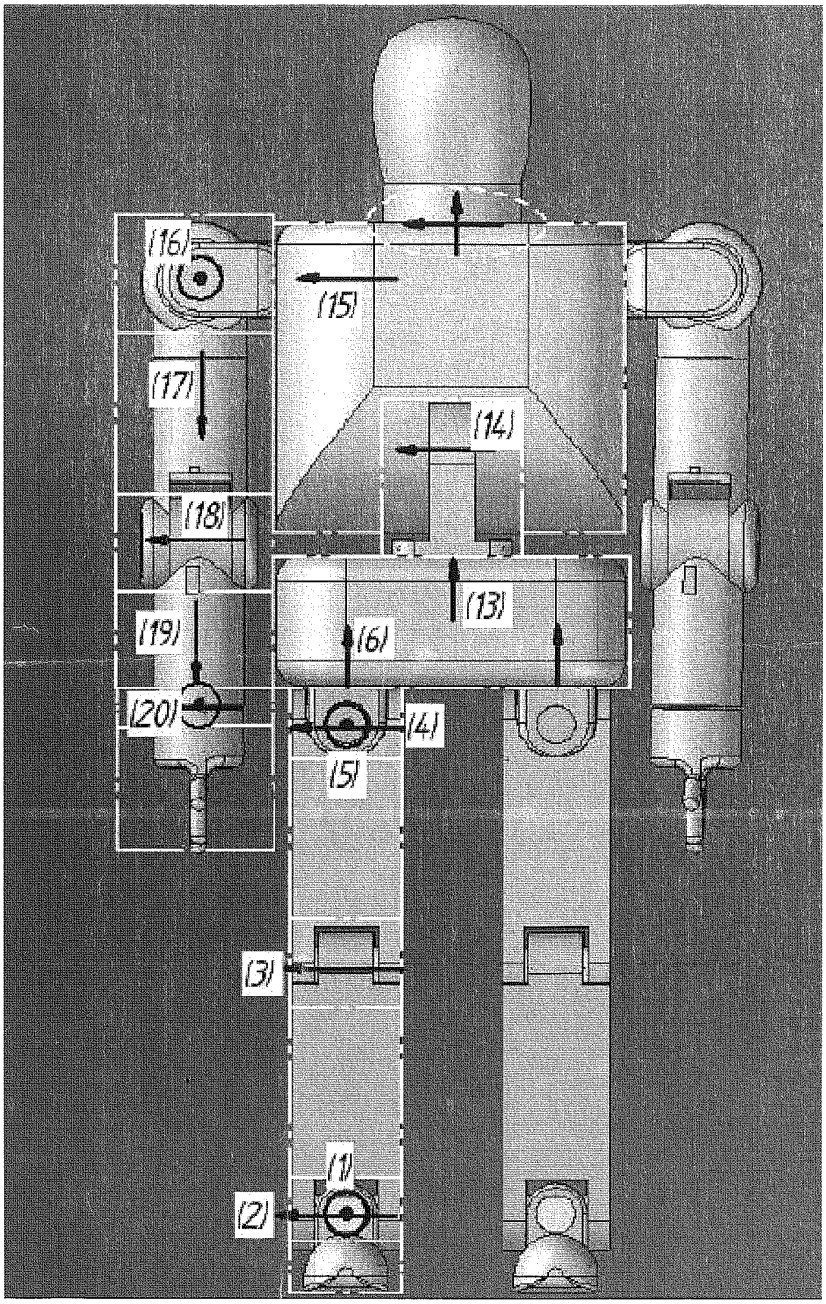


Figura 3.2 Distribución de GDL del robot Rh2

3.2 Modelo VRML del robot Rh-2

3.2.1 Lenguaje VRML

El lenguaje VRML es el que se utiliza en el simulador OpenHRP3 para la representación virtual que realiza dicha plataforma. Para el modelo desarrollado del robot Rh-2 se ha utilizado la versión 2.0 de dicho lenguaje. El modelo se muestra en la Figura 3.3.

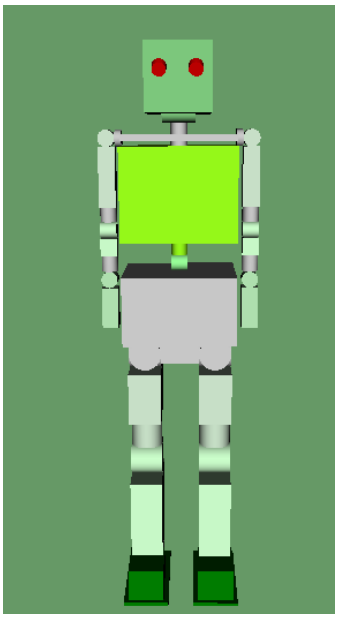


Figura 3.3 Modelo VRML robot Rh-2

3.2.1.1 Historia

El lenguaje VRML [6] surgió en la primavera de 1994 para tratar de acercar los desarrollos de realidad virtual a Internet. Sus creadores llegaron a la conclusión de que se tenía que desarrollar un lenguaje común para la descripción de los mundos en tres dimensiones. De este modo, en la Primera Conferencia Mundial de la World Wide Web, en Ginebra se aprobó el desarrollo de un nuevo lenguaje que permitiese crear mundos en tres dimensiones a los que se pudiera acceder por la World Wide Web.

Ante el amplio apoyo recibido en dicha conferencia y junto a la ayuda de la revista Wired, se estableció una lista de correo con el objetivo de conseguir una primera especificación del lenguaje en unos cinco meses, de forma que se pudiese presentar en la segunda conferencia Web en Octubre de ese mismo año. Tras una serie de debates en los que se discutieron diferentes propuestas, la alternativa presentada por

Silicon Graphics fue la que consiguió un mayor número de votos, convirtiéndose de este modo en la base del nuevo estándar. Esta propuesta consistía en utilizar como punto de partida el lenguaje en el que estaba basada Inventor, un producto de dicha compañía. Finalmente, en Octubre de 1994 se presentó la especificación de VRML 1.0.

VRML 1.0 es un lenguaje para la descripción de mundos virtuales estáticos que cumple tres requisitos fundamentales:

- Es independiente de la plataforma donde se ejecute el visualizador.
- Tiene capacidad para trabajar de un modo eficiente con conexiones lentas.
- Extensibilidad, es decir, facilidad para futuras ampliaciones del lenguaje.

Sin embargo, tras el tiempo de chequeo inicial, se observó que los mundos estáticos no eran suficientes, sino que hacía falta que los objetos tuviesen comportamientos propios y que el usuario pudiese interactuar con ellos, esto llevó a la versión 2.0 (también conocida como VRML 97). Esta nueva versión es mucho más sofisticada que su predecesora y en la cual destacan los siguientes aspectos:

- **MEJORA EN LA CREACIÓN DE MUNDOS ESTÁTICOS**

  Usando las nuevas características de VRML97, se puede añadir realismo a la geometría estática de un mundo. Los nuevos nodos permiten crear fondos para las escenas, añadir montañas y nubes, así como dimensionar objetos con niebla. Otros nuevos nodos permiten crear fácilmente terreno irregular en lugar de usar planos uniformes para superficies de tierra.

  Provee de nodos de generación de sonido espacial 3D para lograr mayor realismo. Se pueden poner vasos que se rompen, teléfonos que suenan o cualquier otro sonido dentro de la escena.

  Los ficheros de optimización y análisis (Parser) son más sencillos que en VRML 1.0, gracias a una nueva estructura de escena grafica simplificada.

- **INTERACCION**

  El movimiento ya no será como si estuviéramos en un mundo muerto y frio. Ahora se va a poder interactuar directamente con objetos y elementos que se encuentren dentro del entorno. Se crean nuevos nodos "*Sensor*" que detectan los eventos provocados por el movimiento en ciertas áreas de un mundo y por el contacto con ciertos objetos, permitiendo incluso trasladar dichos objetos o controles de un lugar a otro. Otra clase de *Sensor* controla el paso del tiempo, proveyendo desde las alarmas de reloj hasta las animaciones repetitivas.

  Se resuelve el problema de la versión 1.0 de caminar a través de las paredes cuando te acercas a algún objeto. La detección de colisiones (nodo "*Collision*") asegura que los objetos sólidos van a reaccionar como tales. El terreno sigue permitiendo viajar subiendo y bajando escaleras o rampas.

- **ANIMACIÓN**

  VRML 2.0 incluye una variedad de objetos animados llamado "*Interpolator*", lo cual permite crear animaciones predefinidas de muchos aspectos del mundo y ejecutarlas en el momento oportuno. Con los interpoladores de movimiento se pueden crear objetos que se mueven, como pájaros volando, puertas que se abren automáticamente o robots que andan. Objetos que cambian de color al moverse, como el sol, objetos que moldean su geometría de una forma a otra, e incluso se pueden crear zooms guiados que automáticamente mueven al usuario a través de un directorio predefinido.

- **SCRIPTING**

  VRML 2.0 no debería ser capaz de moverse sin los nuevos nodos "*Script*". El uso de los scripts permite no solo la animación de criaturas y elementos en un mundo, sino que además les proporciona un toque de inteligencia. Perros animados que puedan buscar periódicos, las manillas del reloj se pueden mover, pájaros que puedan volar, robots que pueden hacer juegos malabares, etc.

  Un script toma una entrada de los sensores y genera eventos basados en esa entrada. Los eventos son pasados a través de diversos nodos por unas guías especiales llamadas "*Routes*".

- **PROTOTIPOS**

  En VRML 2.0 se pueden agrupar un conjunto de nodos como un nuevo tipo de nodo. Esto se denomina prototipo (“*Prototype*”), y una vez desarrollado, éste prototipo se encontrará disponible para cualquiera que quiera usarlo. Se pueden crear instancias de estos prototipos con diferentes valores. Por ejemplo, se puede crear un prototipo de robot con un campo “RobotColor” y entonces crear muchos nodos robot diferentes, coloreados de diferentes maneras.

3.2.2 Estructura del fichero VRML

El fichero VRML desarrollado para el modelo en el simulador OpenHRP3 [8] consta de unos elementos básicos:

- Cabecera
- Comentarios
- Nodos

La cabecera indica el estándar empleado, la versión del archivo VRML y el uso de caracteres internacionales. En este caso la cabecera del archivo es: #VRML V2.0 utf8. Es importante resaltar que no debe existir ningún espacio en blanco entre el símbolo "#" y la palabra "VRML".

Los comentarios en VRML se escriben en una sola línea, la cual comienza con el símbolo "#". Se pueden incluir tantas líneas de comentarios como se desee.

Un nodo es la estructura mínima indivisible de un fichero VRML y tiene como misión la definición de las características de un objeto o bien las relaciones entre distintos objetos. Los nodos contienen campos que describen propiedades de los objetos. Todo campo tiene un tipo determinado y no se puede inicializar con valores de otro tipo. De este modo, cada tipo de nodo tiene una serie de valores predeterminados para todos sus campos, de forma que cuando se utilice solo debe indicarse aquellos campos que se quieran modificar. Los campos pueden ser simples o campos que indiquen a vectores u otros nodos.

Se debe destacar que la estructura de programación usando el lenguaje VRML 2.0 siempre tendrá las mismas especificaciones:

- VRML es un lenguaje sensible a mayúsculas y minúsculas, lo cual ha de ser tenido en cuenta a la hora de asignar nombres.
- Todos los nodos han de comenzar siempre con letra mayúscula.
- Los campos de los nodos deben comenzar siempre con letra minúscula.
- Los números se escriben en punto flotante.
- Utilizar una línea distinta para cada nodo, para cada campo y para cada valor de cada campo.
- Recurrir a la tabulación de cada línea, según su jerarquía.
- Colocar cada símbolo de cierre en el nivel de tabulación que le corresponda.
- Poner las líneas de comentario necesarias al mismo nivel que lo que se comenta.
- Poniendo nombres propios a los nodos

3.2.3 Lenguaje VRML en OpenHRP3

En OpenHRP3, el modelo se define mediante el uso de VRML 2.0 que se utiliza para definir modelos en 3 dimensiones. Cada entidad creada requiere un archivo de modelos individuales. Lo que significa, para expresar una situación de un robot de pie en el suelo, se necesitan dos archivos VRML, una para el suelo y otro para el robot, lo que haría dos archivos VRML en total.

De ahora en adelante, se describe cómo crear un archivo de modelo. En OpenHRP3 se utiliza un nodo denominado "proto", que es similar al concepto de la estructura del lenguaje C. La disposición básica del fichero es la siguiente:

- **Cabecera:**
  PROTO parte de declaración (declaración de la estructura PROTO).
- **Resto de la declaración:**
  Parte de la definición del modelo real (Declaración de las instancias usadas por PROTO).

El esquema básico de definición de las partes del modelo real es el siguiente:

```
Humanoid sample (El nodo raíz de todo el modelo)
     + Joint WAIST (El centro del robot. Un punto libre en el
espacio)
     |  ....
     |   + Joint Chest
     |     + Joint Head
     |     + Joint Left Arm
     |     + Joint Right Arm
     |
     + Joint Left Leg
     |
     + Joint Right Leg
```

Figura 3.4 Esquema básico de definiciones de un robot

Como se puede ver, consta básicamente de tres partes principales que están unidas. "Left Leg", “Right Leg" y la parte superior del cuerpo que empieza desde "WAIST" están conectados con un punto que no es fijo en el espacio. Así como “Left Arm”, “Rigt Arm” y “Head” están unidos a “Chest”, completando la parte superior del cuerpo.

3.2.3.1 Nodo PROTO

En OpenHRP3, el modelo se realiza mediante la unión de distintos nodos PROTO. El nodo PROTO que se utiliza está basado en el nodo PROTO promulgado por el H-Anim (Humanoid Animation Working Group). Este nodo sigue las especificaciones del H-Anim 1.1, que se utilizó originalmente para describir figuras humanas y que ha sido ampliado y modificado para ser usado en el modelo OpenHRP3. Estos nodos OpenHRP3 PROTO que se utilizarán en el modelo son Humanoid, Joint y Segment.

Además de los tres nodos PROTO mencionados anteriormente, también existen otros nodos PROTO para definir distintos sensores. Se pueden simular varios sensores incluyendo las definiciones de estos sensores, como Gyro sensor, sensor de aceleración, sensor de presión, etc. Actualmente también se incluyen nodos de sensores de visión para simular una cámara y sensores de fuerza para simular fuerzas y pares de fuerzas.

Los nodos utilizados en OpenHRP3 se clasifican de la siguiente forma:

- Nodos que definen la estructura de vínculos y los parámetros dinámicos y mecánicos.
  - Joint Node.
  - Segment Node.
  - Humanoid Node.

- Nodos que definen sensores.
  - VisionSensor Node.
  - ForceSensor Node.
  - Gyro Node.
  - AccelerationSensor Node.
  - PressureSensor Node.

A continuación se explica brevemente cada uno de los nodos utilizados para la creación del modelo del robot Rh-2.

3.2.3.1.1 ***Nodo Joint***

El nodo Joint especifica la estructura de vínculos, es decir, define cada articulación del robot (si son nodos de rotación, de tipo transform, etc) al que irán asociados los eslabones correspondientes definidos en el nodo Segment (que será explicado más adelante). En la Figura 3.5 se muestra la estructura del nodo Joint con cada uno de los campos que lo componen:

```
PROTO Joint [
exposedField     SFVec3f      center              0 0 0
exposedField     MFNode       children            []
exposedField     MFFloat      llimit              []
exposedField     MFFloat      lvlimit             []
exposedField     SFRotation   limitOrientation    0 0 1 0
exposedField     SFString     name                ""
exposedField     SFRotation   rotation            0 0 1 0
exposedField     SFVec3f      scale               1 1 1
exposedField     SFRotation   scaleOrientation    0 0 1 0
exposedField     MFFloat      stiffness           [ 0 0 0 ]
exposedField     SFVec3f      translation         0 0 0
exposedField     MFFloat      ulimit              []
exposedField     MFFloat      uvlimit             []
exposedField     SFString     jointType           ""
exposedField     SFInt32      jointId             -1
exposedField     SFVec3f      jointAxis           0 0 1

exposedField     SFFloat      gearRatio           1
exposedField     SFFloat      rotorInertia        0
exposedField     SFFloat      rotorResistor       0
exposedField     SFFloat      torqueConst         1
exposedField     SFFloat      encoderPulse        1
]
{
Transform {
center           IS center
children         IS children
rotation         IS rotation
scale            IS scale
scaleOrientation IS scaleOrientation
translation      IS translation
}
}
```

Figura 3.5 Estructura del nodo Joint

A continuación se explica cada uno de los campos que componen el nodo Joint.

- **center:** Centro de los ejes de rotación de la articulación. Se puede especificar como un valor de desplazamiento del sistema en relación con el origen de coordenadas local.
- **children:** Utilizado para especificar los nodos hijos (nodos secundarios).
- **llimit:** Límite inferior del valor del ángulo de rotación de la articulación [rad].
- **lvlimit:** Límite inferior del valor de la velocidad angular de rotación de la articulación [rad/seg].
- **limitOrientation:** No utilizado en OpenHRP3.

- **name:** Nombre de la articulación.
- **rotation:** Es la orientación del sistema de coordenadas local. Especifica el valor de rotación [rad], en relación con el nodo padre (nodo anterior).
- **scale:** Permite establecer una escala.
- **scaleOrientation:** Utilizado para especificar la orientación de los sistemas de coordenadas utilizados en la escala.
- **stiffness:** No utilizado en OpenHRP3.
- **translation:** Es la posición del sistema de coordenadas local. Especifica el valor de desplazamiento (traslación en [m]), en relación con el nodo padre (nodo anterior).
- **ulimit:** Límite superior del valor del ángulo de rotación de la articulación [rad].
- **uvlimit:** Límite superior del valor de la velocidad angular de rotación de la articulación [rad/seg].
- **jointType:** Especifica el tipo de articulación. Puede ser de varios tipos:
    - free: Permite el desplazamiento y rotación a lo largo de los 3 ejes de coordenadas (6 grados de libertad).
    - slide: Permite el desplazamiento (traslación) únicamente a lo largo del eje de coordenadas especificado en "jointAxis" (1 grado de libertad).
    - rotate: Permite la rotación sólo alrededor del eje de coordenadas especificado en "jointAxis" (1 grado de libertad).
    - fixed: La articulación no permite ni la traslación, ni la rotación (0 grados de libertad).
- **jointId:** Especifica el número de identificación de la articulación.
- **jointAxis:** Especifica el eje de referencia de la articulación.
- **gearRatio:** Si la razón de la desaceleración del motor de la articulación es 1/100, asignar como 100.
- **gearEfficiency:** Eficiencia de desaceleración. Si el rendimiento es del 60%, asignar como 0.6.
- **rotorInertia:** Especifica el momento de inercia del rotor del motor [kg / m^2].
- **rotorResistor:** Resistencia de la bobina del motor [Ohm].
- **torqueConst:** Constante del par de fuerza [Nm / seg].
- **encoderPulse:** Contador de pulsos del encoder.

3.2.3.1.2 *Nodo Segment*

El nodo Segment define la forma de la pieza que forma la articulación (eslabón). En la Figura 3.6 se muestra la estructura del nodo Segment con cada uno de los campos que lo componen:

```
PROTO Segment [
  field         SFVec3f   bboxCenter        0 0 0
  field         SFVec3f   bboxSize          -1 -1 -1
  exposedField  SFVec3f   centerOfMass      0 0 0
  exposedField  MFNode    children          [ ]
  exposedField  SFNode    coord             NULL
  exposedField  MFNode    displacers        [ ]
  exposedField  SFFloat   mass              0
  exposedField  MFFloat   momentsOfInertia  [ 0 0 0 0 0 0 0 0 0 ]
  exposedField  SFString  name              ""
  eventIn       MFNode    addChildren
  eventIn       MFNode    removeChildren
]
{
  Group {
    addChildren    IS addChildren
    bboxCenter     IS bboxCenter
    bboxSize       IS bboxSize
    children       IS children
    removeChildren IS removeChildren
  }
}
```

Figura 3.6 Estructura del nodo Segment

A continuación se explica cada uno de los campos que componen el nodo Segment:

- **bboxCenter:** No utilizado en OpenHRP3.
- **bboxSize:** No utilizado en OpenHRP3.
- **centerOfMass:** Coordenadas del centro de gravedad.
- **children:** Especifica los nodos secundarios, se añaden los nodos que definen la forma.
- **coord:** No utilizado en OpenHRP3.
- **displacers:** No utilizado en OpenHRP3.
- **mass:** Masa de la articulación.
- **momentsOfInertia:** Momento de inercia de la articulación.
- **name:** Nombre de la articulación.
- **addChildren:** No utilizado en OpenHRP3.
- **removeChildren:** No utilizado en OpenHRP3.

3.2.3.1.3 *Nodo Humanoid*

El nodo Humanoid es el nodo raíz del modelo. En la Figura 3.7 se muestra la estructura de dicho nodo con cada uno de los campos que lo componen:

```
PROTO Humanoid [
  field         SFVec3f     bboxCenter        0 0 0
  field         SFVec3f     bboxSize          -1 -1 -1
  exposedField  SFVec3f     center            0 0 0
  exposedField  MFNode      humanoidBody      [ ]
  exposedField  MFString    info              [ ]
  exposedField  MFNode      joints            [ ]
  exposedField  SFString    name              ""
  exposedField  SFRotation  rotation          0 0 1 0
  exposedField  SFVec3f     scale             1 1 1
  exposedField  SFRotation  scaleOrientation  0 0 1 0
  exposedField  MFNode      segments          [ ]
  exposedField  MFNode      sites             [ ]
  exposedField  SFVec3f     translation       0 0 0
  exposedField  SFString    version           "1.1"
  exposedField  MFNode      viewpoints        [ ]
]
{
  Transform {
    bboxCenter       IS bboxCenter
    bboxSize         IS bboxSize
    center           IS center
    rotation         IS rotation
    scale            IS scale
    scaleOrientation IS scaleOrientation
    translation      IS translation
    children [
      Group {
        children IS viewpoints
      }
      Group {
        children IS humanoidBody
      }
    ]
  }
}
```

Figura 3.7 Estructura del nodo Humanoid

A continuación se explica cada uno de los campos que componen el nodo Humanoid:

- **bboxCenter:** No utilizado en OpenHRP3.
- **bboxSize:** No utilizado en OpenHRP3.
- **center:** Explicado en el punto 3.2.3.1.1.
- **humanoidBody:** Utilizado para especificar los nodos secundarios dentro del cuerpo del Humanoide.

- **info:** Este campo se utiliza para realizar comentarios sobre el modelo.
- **joints:** Almacena la lista de los nodos Joint definidos.
- **name:** Especifica el nombre del modelo.
- **rotation:** Explicado en el punto 3.2.3.1.1.
- **scale:** Explicado en el punto 3.2.3.1.1.
- **scaleOrientation:** Explicado en el punto 3.2.3.1.1.
- **segments:** Almacena la lista de los nodos Segment definidos.
- **sites:** No utilizado en OpenHRP3.
- **translation:** Explicado en el punto 3.2.3.1.1.
- **version:** Número de versión del modelo.
- **viewpoints:** Posiciones de observación en el entorno virtual.

3.2.3.1.4 *Nodo VisionSensor*

El nodo VisionSensor se utiliza para definir los sensores de visión (simulación del uso de cámaras). En la Figura 3.7 se muestra la estructura del nodo VisionSensor con cada uno de los campos que lo componen:

```
PROTO VisionSensor
[
  exposedField  SFVec3f     translation       0 0 0
  exposedField  SFRotation  rotation          0 0 1 0
  exposedField  SFFloat     fieldOfView       0.785398
  field         SFString    name              ""
  exposedField  SFFloat     frontClipDistance 0.01
  exposedField  SFFloat     backClipDistance  10.0
  exposedField  SFString    type              "NONE"
  exposedField  SFInt32     sensorId          -1
  exposedField  SFInt32     width             320
  exposedField  SFInt32     height            240
]
{
  Transform
  {
    translation IS translation
    rotation    IS rotation
  }
}
```

Figura 3.8 Estructura del nodo VisionSensor

A continuación se explica cada uno de los campos que componen el nodo VisionSensor:

- **translation:** Posición del sistema de coordenadas local. Especifica el valor de desplazamiento (traslación en [m]), en relación con el sistema de coordenadas del nodo anterior (nodo padre).
- **rotation:** Orientación del sistema de coordenadas local. Especifica el valor de rotación [rad], en relación con el sistema de coordenadas del nodo padre.
- **fieldOfView:** Ángulo de visión de la cámara [rad]. Rango válido entre 0 y pi.
- **name:** Nombre del sensor.
- **frontClipDistante:** Distancia desde el punto de observación a la parte delantera.
- **backClipDistance:** Distancia desde el punto de observación a la parte trasera.
- **type:** Especifica los tipos de información que pueden ser adquiridos por el sensor. Estos tipos de datos pueden ser:
    - color: Información sobre el color del producto.
    - depth: Información sobre la profundidad del producto.
    - color_depth: Información sobre el color y la profundidad del producto.
    - none: No especifica ninguna información sobre el producto.
- **sensorId:** Número de identificación del sensor.
- **width:** Especifica el ancho de la imagen.
- **height:** Especifica la altura de la imagen.

3.2.3.1.5 *Nodo ForceSensor*

El nodo ForceSensor se utiliza para definir los sensores de fuerza y de pares de fuerza. En la Figura 3.9 se muestra la estructura del nodo ForceSensor con cada uno de los campos que lo componen:

```
PROTO ForceSensor [
  exposedField SFVec3f maxForce -1 -1 -1
  exposedField SFVec3f maxTorque -1 -1 -1
  exposedField SFVec3f translation 0 0 0
  exposedField SFRotation rotation 0 0 1 0
  exposedField SFInt32 sensorId -1
]
{
  Transform {
translation IS translation
    rotation IS rotation
  }
}
```

Figura 3.9 Estructura del nodo ForceSensor

A continuación se explica cada uno de los campos que componen el nodo ForceSensor:

- **maxForce**: Fuerza máxima que puede ser medida por el sensor.
- **maxTorque**: Par máximo que puede ser medido por el sensor.
- **translation**: Posición del sistema de coordenadas local. Especifica el valor de desplazamiento, en relación con el sistema de coordenadas del nodo padre.
- **rotation**: Orientación del sistema de coordenadas local. Especifica el valor de rotación, en relación con el sistema de coordenadas del nodo padre.
- **sensorId**: Número de identificación del sensor.

3.2.4 Partes del fichero VRML para el modelo del robot Rh-2

El nodo Humanoid ([2] y [8]) está definido como el nodo raíz del modelo, y sólo puede existir un único nodo Humanoid para cada modelo.

Las articulaciones se definen mediante el nodo Joint. Este nodo contiene la información del enlace de esta articulación con la anterior. En la Figura 3.10 se muestra la estructura de un nodo Joint del modelo del robot Rh-2 usado en OpenHRP3. En este caso se trata del nodo Joint que define un de los GDL que existe en el hombro del modelo, una articulación que tiene como eje de rotación el eje Y.

```
DEF LARM_SHOULDER_P Joint {
                      jointType "rotate"
                    jointAxis "Y"
                      jointId  18
                      translation   0 0.215 -0.005
                      children [
                        DEF LARM_LINK1 Segment {
                          centerOfMass  0.1 0 0
                          mass    3.0
                          momentsOfInertia [ 1 0 0 0 1 0 0 0 1 ]
                          DEF sensor6 ForceSensor { sensorId 6 }
                          children Transform {
                            translation 0 -0.035 0
                            children DEF ARM_SHAPE1 Shape {
                              appearance Appearance {
                                material Material {
                                }
                              }
                              geometry Cylinder { radius 0.025
height 0.02 }
                            }
                          }
                        }
```

Figura 3.10 Estructura de definición de un nodo Joint

A continuación se analiza la estructura del código de dicha definición:

En la primera línea se muestra la estructura de la sentencia que define un nodo Joint, en este caso el nodo llamado LARM_SHOULDER_P:

***DEF [NOMBRE JOINT] Joint* {**

En la segunda línea se especifica el tipo de articulación, que como se vio en el punto 3.2.3.1.1 puede ser cuatro de cuatro tipos: free, rotate, slide y fixed. En este caso se trata de una articulación de tipo "rotate". La estructura que se utiliza para esta definición es la siguiente:

jointType "[tipo]"

Posteriormente se indica respecto a qué eje rota o traslada (en función del tipo de articulación de la que se trate). Para este caso es el eje "Y". Este campo se define de la siguiente manera:

jointAxis "[eje]"

El sistema de coordenadas del archivo VRML en OpenHRP3 se muestra en la Figura 3.11. Este sistema de coordenadas se rige por la regla de la mano derecha.

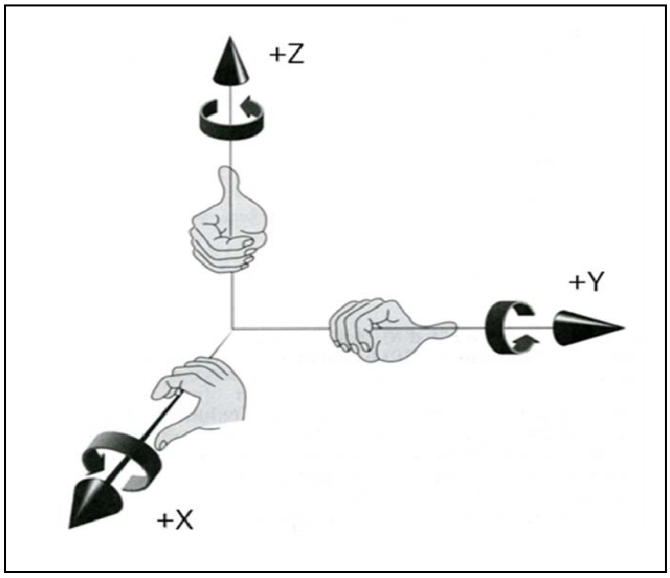


Figura 3.11 Sistema de coordenadas VRML en OpenHRP

A continuación se indica el número identificación de la articulación. El campo "JointId" debe ser definido de acuerdo a las siguientes reglas:

- "JointId" debe empezar desde 0 (cero).
- "JointId" debe ser una secuencia continua de números enteros que no tiene espacios.
- La articulación con identificador 0 debe ser del tipo "fixed", porque es el punto de referencia del robot.

La estructura para definir el número de identificación de la articulación es la siguiente:

jointId [número]

Después indicamos la traslación de la articulación respecto a la anterior articulación, mencionar que las unidades por las que se rige este campo son metros:

translation [x] [y] [z]

Tras definir los campos necesarios en el nodo Joint, se define un sensor de fuerza, porque interesa conocer las fuerzas, pero sobre todo, los pares de fuerzas que soporta el robot en ese nodo (el hombro), esto será imprescindible para su diseño.
Para incluir un sensor de fuerza, el nodo ForceSensor debe ser incluido a continuación de la definición de un nodo segment. Por ejemplo, en nuestro caso, se debe definir como se muestra en la Figura 3.12:

```
DEF LARM_SHOULDER_P Joint
{
  children
  [
    DEF LARM_LINK1 Segment {
        children[
            DEF FORCE_SENSOR ForceSensor
            {
              :
              :
              :
            }
        ]
    }
  ]
}
```

Figura 3.12 Definición nodo ForceSensor

La estructura para definir el número de identificación del sensor es la siguiente:

DEF [nombre del sensor] ForceSensor { sensorId[número del sensor] }

A continuación, se pasa a definir el nodo Segment, en el cual se encuentran los campos que muestran la apariencia de la articulación definida anteriormente.

En primer lugar se debe indicar el centro de masas de la pieza, la masa y su momento de inercia. Posteriormente, a través del nodo Transform, se define la apariencia física de la pieza.

En este caso la pieza es desplazada -0.035 m respecto del eje Y, a través del campo translation.

translation [x] [y] [z]

Tras esto se utiliza el nodo Shape para definir tanto la geometría como la apariencia. Dentro del nodo Appearance se describen las propiedades del material (color, transparencia, textura, etc.). Algunos de los campos del nodo Material son los siguientes:

- *diffuse color* - color principal de sombreado
- *emissive color* - color del brillo
- *transparency* – si es transparente u opaco

Mencionar que los colores en VRML se definen como RGB (Red, Green, Blue) con valores comprendidos en el intervalo 0.0 (nada) y 1.0 (todo). En la Tabla 3.1 se muestra algunos de los valores RGB en VRML para definir diferentes colores.

Color	R	G	B
Blanco	1.0	1.0	1.0
Negro	0.0	0.0	0.0
Rojo	1.0	0.0	0.0
Verde	0.0	1.0	0.0
Azul	0.0	0.0	1.0
Amarillo	1.0	1.0	0.0
Magenta	1.0	0.0	1.0
Marrón	0.5	0.2	0.0

Tabla 3.1. Valores RGB para diferentes colores en VRML

Por último, se define el campo *geometry,* en el cual se pueden utilizar distintas figuras geométricas primitivas. Estas figuras geométricas son las siguientes:

- Box (caja)
- Cone (cono)
- Cylinder (cilindro)
- Sphere (esfera)

Las formas Box y Cylinder serán las más utilizadas a la hora de modelar el humanoide.

Por otra parte, se deben definir los demás sensores (aparte del sensor de fuerza definido anteriormente) que lleva incluidos el modelo. En primer lugar se explica la definición del nodo VisionSensor, que corresponde con los sensores de visión para la simulación de una cámara.

El nodo VisionSensor debe ser incluido debajo de un nodo Joint. Pero si existe un nodo Segment dentro del nodo Joint al cual se quiere incluir el sensor de visión, primero se debe definir el nodo Segment y a continuación el nodo VisionSensor. En la Figura 3.13 se muestra un ejemplo sobre cómo definir un nodo VisionSensor.

```
DEF CHEST_Y Joint
{
  children
  [
    DEF WAIST_LINK2 Segment

    {
         :
         :
         :
    }

    DEF VISION_SENSOR VisionSensor
    {
         :
         :
         :
    }
  ]
}
```

Figura 3.13 Definición nodo VisionSensor

3.2.5 Estructura general del fichero VRML para el robot Rh-2

El fichero VRML del modelo del robot Rh-2 ha sido diseñado de acuerdo con los parámetros originales del robot. Se han respetado todas las medidas de cada una de las piezas que forman el robot, así como las masas esperadas de cada una de las articulaciones, para conseguir un modelo lo más parecido en todos los aspectos a los planos proporcionados del robot real. La documentación con la que se ha realizado el modelo se podrá comprobar en el anexo de este proyecto.

En la Figura 3.14 se muestra el modelo final del robot en el simulador, en la cual aparecen señaladas cada una de las articulaciones con sus respectivos nombres, donde se puede comprobar que el modelo consta de 28 GDL como se comentó en la descripción del robot.

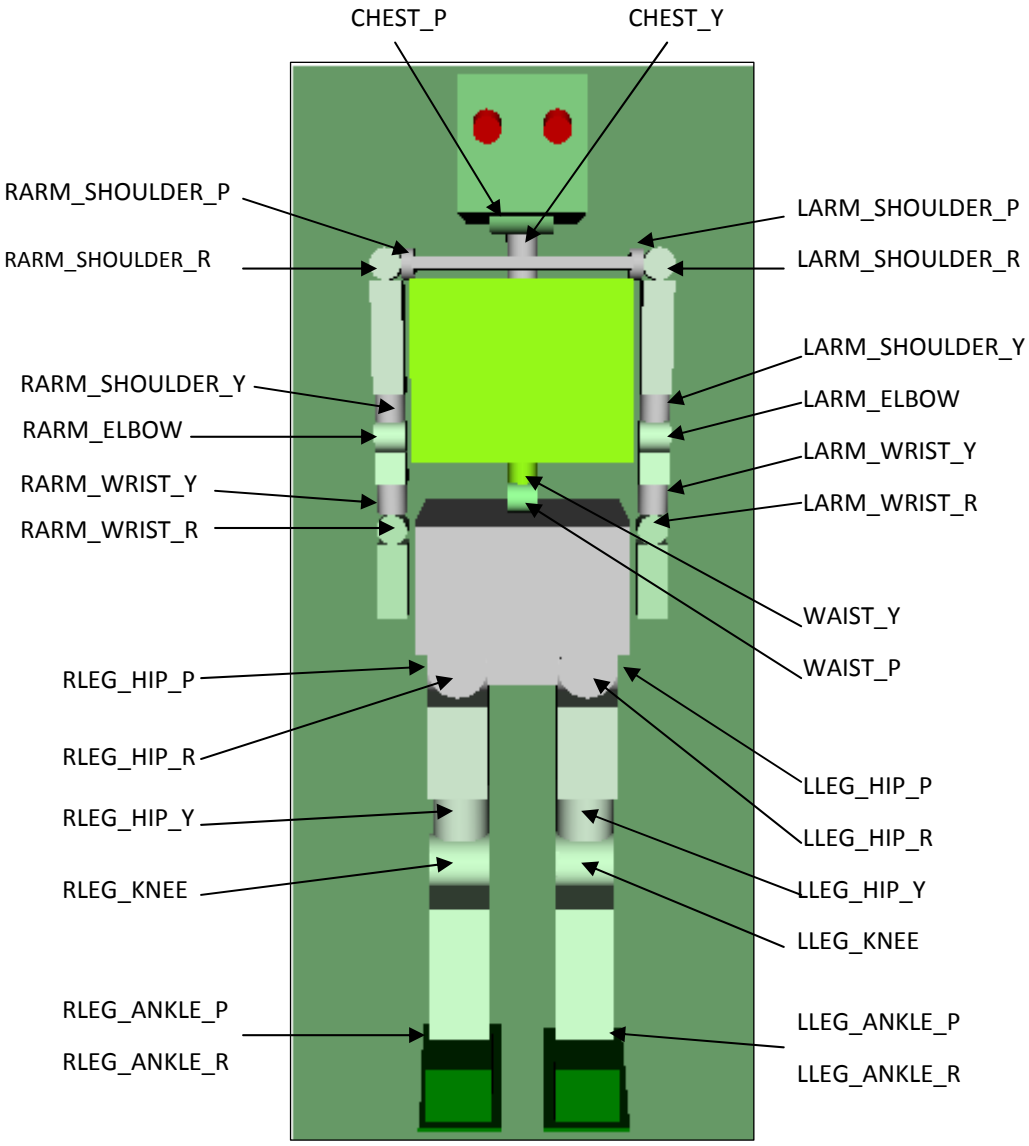


Figura 3.14 Modelo VRML del robot Rh-2 con sus respectivas articulaciones

A continuación se muestra la estructura de programación del modelo mediante su código VRML (Figura 3.15). En él se puede apreciar la estructura de las relaciones entre articulaciones del modelo. Se debe mencionar que la articulación principal del modelo es la cintura (WAIST_Y), a partir de ella se construye la parte superior del cuerpo, seguido de los brazos y en último lugar las piernas.

```
+-humanoidBody
  |
  | # Parte superior del cuerpo
  +-Joint WAIST : Segment WAIST_LINK0
  |   Joint WAIST_P : Segment WAIST_LINK1
  |     Joint WAIST_Y : Segment WAIST_LINK2
  |       Joint CHEST_Y : Segment WAIST_LINK3
  |         Joint CHEST_P : Segment WAIST_LINK4
  |         | # Cámaras
  |         +-VisionSensor VISION_SENSOR1
  |         +-VisionSensor VISION_SENSOR2
  |
  | # Brazo izquierdo
  +-Joint LARM_SHOULDER_P : Segment LARM_LINK1
  |   Joint LARM_SHOULDER_R : Segment LARM_LINK2
  |     Joint LARM_SHOULDER_Y : Segment LARM_LINK3
  |       Joint LARM_ELBOW : Segment LARM_LINK4
  |         Joint LARM_WRIST_Y : Segment LARM_LINK5
  |           Joint LARM_WRIST_R : Segment LARM_LINK6
  |
  |
  | # Brazo derecho
  +-Joint RARM_SHOULDER_P : Segment RARM_LINK1
  |   Joint RARM_SHOULDER_R : Segment RARM_LINK2
  |     Joint RARM_SHOULDER_Y : Segment RARM_LINK3
  |       Joint RARM_ELBOW : Segment RARM_LINK4
  |         Joint RARM_WRIST_Y : Segment RARM_LINK5
  |           Joint RARM_WRIST_R : Segment RARM_LINK6
  |
  |
  | # Pierna izquierda
  +-Joint LLEG_HIP_R : Segment LLEG_LINK1
  |   Joint LLEG_HIP_P : Segment LLEG_LINK2
  |     |
  |     | # Sensor de fuerza
  |     +-ForceSensor sensor19
  |       Joint LLEG_HIP_Y : Segment LLEG_LINK3
  |         Joint LLEG_KNEE : Segment LLEG_LINK4
  |          |
  |          | # Sensor de fuerza
  |          +-ForceSensor sensor21
  |            Joint LLEG_ANKLE_P : Segment LLEG_LINK5
  |            |
  |            | # Sensor de fuerza
  |            +-ForceSensor sensor22
  |            Joint LLEG_ANKLE_R : Segment LLEG_LINK6
  |
  | # Pierna derecha
  +-Joint RLEG_HIP_R : Segment RLEG_LINK1
  |   Joint RLEG_HIP_P : Segment RLEG_LINK2
  |     |
  |     | # Sensor de fuerza
  |     +-ForceSensor sensor25
  |       Joint RLEG_HIP_Y : Segment RLEG_LINK3
  |         Joint RLEG_KNEE : Segment RLEG_LINK4
  |          |
  |          | # Sensor de fuerza
  |          +-ForceSensor sensor27
  |            Joint RLEG_ANKLE_P : Segment RLEG_LINK5
  |            |
  |            | # Sensor de fuerza
  |            +-ForceSensor sensor28
  |            Joint RLEG_ANKLE_R : Segment RLEG_LINK6
```

Figura 3.15 Estructura de articulaciones y sensores del modelo VRML

En la Figura 3.16 se muestra el modelo del robot donde se señalan los eslabones (segment) que terminan de completar el modelo.

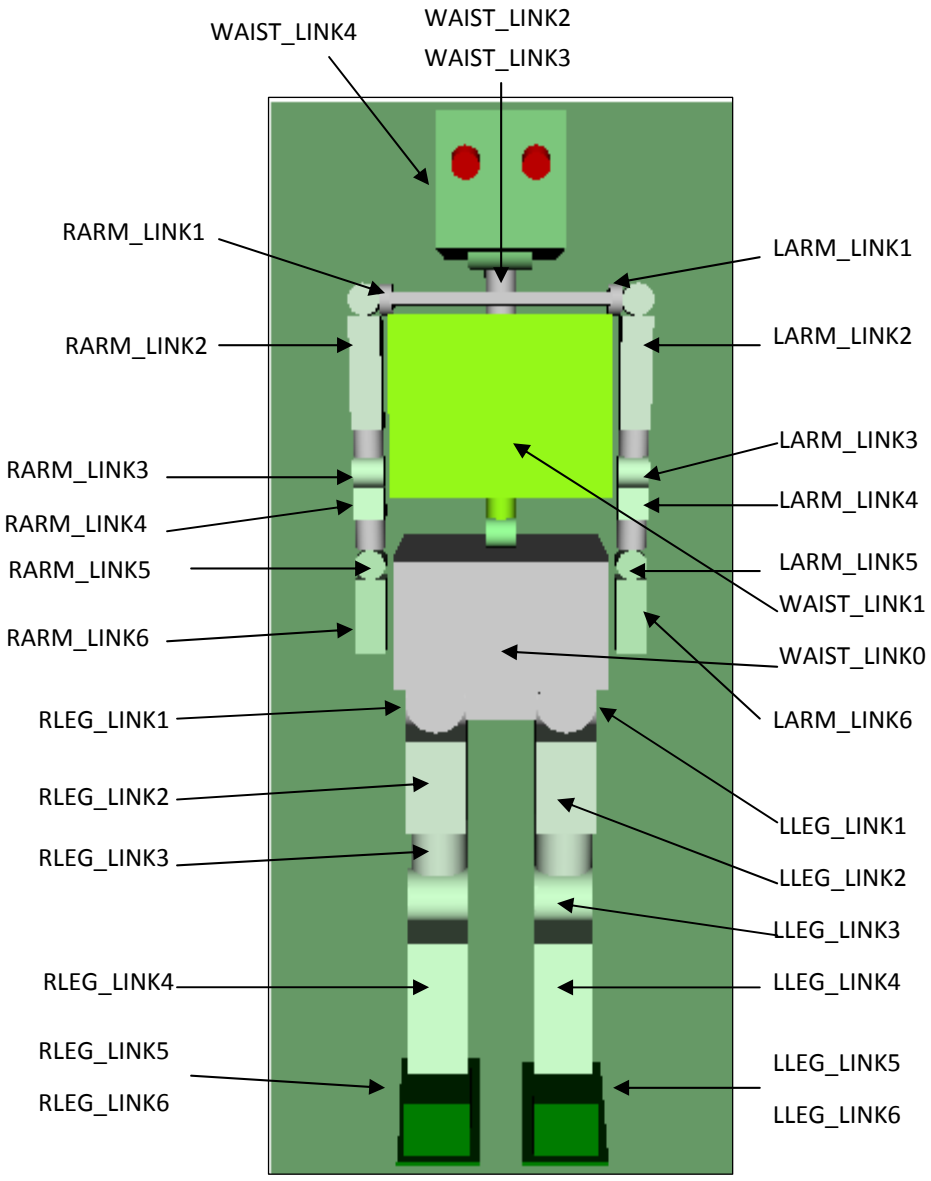


Figura 3.16 Modelo VRML del robot Rh-2 con eslabones

En el Anexo se incluye el código completo del fichero VRML que define el modelo del robot Rh-2 para el simulador OpenHRP3.

4 Simulación de tareas en OpenHRP3

4.1 Archivos necesarios para la simulación

En OpenHRP3 existen determinados archivos que se deben cambiar para llevar a cabo la adaptación del modelo que venía por defecto al modelo que se quiere simular en dicha plataforma.

Uno de estos archivos es el propio modelo VRML del robot, el cual fue descrito en el capítulo anterior. Por otro lado será necesario también adaptar el archivo de trayectorias del robot (posición o angle, velocidad y aceleración de las articulaciones), el archivo que define el controlador (programado en C++) y el archivo del proyecto, donde se carga cada uno de los elementos de la simulación. Todos estos archivos serán definidos a continuación.

4.1.1 Archivos de trayectorias

La trayectoria del robot viene definida por tres archivos, que forman parte del controlador del simulador y están ubicados en la siguiente dirección:
C:\Archivos de programa\OpenHRP-3.0.2\Controller\rtc\Nombre_del_proyecto(para este caso se ha utilizado el proyecto sampleHG)\etc, dentro de la cual se pueden encontrar tres archivos:

- Archivo de posición de las articulaciones (angle.dat) en donde los datos están introducidos en radianes [rad].
- Archivo de velocidad de las articulaciones (vel.dat), donde los datos son introducidos en radianes por segundo [rad/seg].
- Y por último el archivo de aceleración de cada articulación (acc.dat), donde los datos están introducidos en [rad/seg^2].

Estos archivos serán leídos por OpenHRP3 en cada instante de simulación y le indicarán el sentido y la velocidad del movimiento de cada articulación.

Estos archivos tienen extensión *.dat y están estructurados en columnas (ver Figura 4.1). Cada una de las columnas de valores va asociada con una articulación. La primera de estas columnas define la base de tiempos en la que se realiza la simulación.

0	0	0	0	0	0	0	0	0	0	0	0
0.005	0	-7.13E-06	0	1.43E-05	-7.13E-06	0	0	0	0	0	0
0.01	0	-2.84E-05	0	5.69E-05	-2.84E-05	0	0	0	0	0	0
0.015	0	-6.39E-05	0	0.00012774	-6.39E-05	0	0	0	0	0	0
0.02	0	-0.00011332	0	0.00022664	-0.00011332	0	0	0	0	0	0
0.025	0	-0.00017671	0	0.00035342	-0.00017671	0	0	0	0	0	0
0.03	0	-0.00025395	0	0.0005079	-0.00025395	0	0	0	0	0	0
0.035	0	-0.00034495	0	0.00068991	-0.00034495	0	0	0	0	0	0
0.04	0	-0.00044964	0	0.00089928	-0.00044964	0	0	0	0	0	0
0.045	0	-0.00056793	0	0.00113585	-0.00056793	0	0	0	0	0	0
0.05	0	-0.00069972	0	0.00139945	-0.00069972	0	0	0	0	0	0
0.055	0	-0.00084495	0	0.0016899	-0.00084495	0	0	0	0	0	0
0.06	0	-0.00100352	0	0.00200704	-0.00100352	0	0	0	0	0	0
0.065	0	-0.00117535	0	0.0023507	-0.00117535	0	0	0	0	0	0
0.07	0	-0.00136035	0	0.0027207	-0.00136035	0	0	0	0	0	0
0.075	0	-0.00155845	0	0.00311689	-0.00155845	0	0	0	0	0	0
0.08	0	-0.00176955	0	0.0035391	-0.00176955	0	0	0	0	0	0
0.085	0	-0.00199357	0	0.00398715	-0.00199357	0	0	0	0	0	0
0.09	0	-0.00223044	0	0.00446088	-0.00223044	0	0	0	0	0	0
0.095	0	-0.00248006	0	0.00496013	-0.00248006	0	0	0	0	0	0
0.1	0	-0.00274236	0	0.00548472	-0.00274236	0	0	0	0	0	0
0.105	0	-0.00301724	0	0.00603449	-0.00301724	0	0	0	0	0	0
0.11	0	-0.00330464	0	0.00660928	-0.00330464	0	0	0	0	0	0
0.115	0	-0.00360446	0	0.00720891	-0.00360446	0	0	0	0	0	0
0.12	0	-0.00391661	0	0.00783323	-0.00391661	0	0	0	0	0	0

Figura 4.1 Estructura del archivo .dat

La segunda columna corresponde a la articulación con número de identificación 0 (jointId=0). La tercera columna corresponderá a la articulación con número de identificación 1 (jointId=1) y así sucesivamente se irán asociando las siguientes columnas con el resto de articulaciones.

Para las simulaciones en concreto que se realizarán, los archivos contarán con 29 columnas (1 columna para la base de tiempos y 28 columnas por cada uno de los GDL que contiene el modelo).

En las simulaciones realizadas se proporcionaban los datos de posición (angle) y velocidad de las articulaciones, la introducción de estos datos en estos archivos se explicará en el capítulo siguiente.

4.1.2 Archivo del controlador

Además de los tres archivos donde se define la trayectoria, que pertenecen al controlador, también debemos configurar adecuadamente el controlador propiamente dicho (este archivo se encuentra en el directorio: C:\Archivos de programa\OpenHRP-3.0.2\Controller\rtc\Nombre_del_proyecto(para este caso se ha utilizado el proyecto sampleHG)\Nombre_del_proyecto.cpp (en este caso SampleHG.cpp)).

Una de las funciones del controlador es ir leyendo los valores de los archivos donde se encuentran los ángulos, velocidades y aceleraciones para cada instante de tiempo de simulación. Por este motivo debemos seleccionar el valor de la constante DOF que indica el número de GDL que contiene el modelo a simular. En este caso el valor de esta constante debe ser 28. Esta constante se utiliza en el código dentro de un bucle *for* (Figura 4.2) que va recorriendo cada uno de los valores de los tres archivos. Cabe destacar que al modificar este archivo, se debe volver a recompilar en Visual Studio OpenHRP3 para que el cambio surta efecto en la simulación.

```
#define DOF (28)
  .
  .
  .

for (i=0; i<DOF; i++)
  {
      angle >> m_angle.data[i];
      vel   >> m_vel.data[i];
      acc   >> m_acc.data[i];
  }
```

Figura 4.2 Muestra del código del archivo del controlador

4.1.3 Archivos de proyecto

Otro de los archivos necesarios para la simulación es el archivo que configura los elementos de simulación para el proyecto que se va a emplear. Este archivo tiene extensión .XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<grxui>
    <mode name="Simulation">
        <item class="com.generalrobotix.ui.item.GrxWorldStateItem"
name="untitled" select="true">
            <property name="logTimeStep" value="0.005 "/>
            <property name="integrate" value="true"/>
            <property name="viewsimulate" value="false"/>
            <property name="totalTime" value="7.0 "/>
            <property name="timeStep" value="0.005 "/>
            <property name="method" value="RUNGE_KUTTA"/>
            <property name="gravity" value="9.8 "/>
            <property name="viewsimulationTimeStep" value="0.033 "/>
        </item>
        <item class="com.generalrobotix.ui.item.GrxModelItem"
name="floor" select="true" url="$(OPENHRPHOME)/etc/floor.wrl">
            <property name="isRobot" value="false"/>
            <property name="WAIST.rotation" value="0.0 1.0 0.0 0.0 "/>
            <property name="WAIST.translation" value="0.0 0.0 -0.1 "/>
        </item>
      <item class="com.generalrobotix.ui.item.GrxModelItem" name="box3"
select="true" url="$(OPENHRPHOME)/etc/box3.wrl">
        <property name="isRobot" value="false"/>
        <property name="WAIST.rotation" value="0.0 0.0 0.0 0.0 "/>
        <property name="WAIST.translation" value="0.5 0.0 0.15 "/>
      </item>
        <item class="com.generalrobotix.ui.item.GrxModelItem"
name="sample" select="true" url="$(OPENHRPHOME)/etc/sample.wrl">
            <property name="isRobot" value="true"/>
            <property name="controller" value="SampleHGController"/>
            <property name="controlTime" value="0.002"/>
            <property name="setupDirectory"
value="$(OPENHRPHOME)/Controller/rtc/SampleHG"/>
            <property name="setupCommand" value="SampleHG$(BIN_SFX)"/>
            <property name="RLEG_HIP_R.angle" value="0.0 "/>
            <property name="RARM_SHOULDER_R.mode" value="HighGain"/>
            <property name="LLEG_KNEE.mode" value="HighGain"/>
            <property name="RARM_ELBOW.angle" value="-1.5708 "/>
            <property name="LLEG_ANKLE_P.mode" value="HighGain"/>
            <property name="RLEG_ANKLE_P.angle" value="-0.0424675 "/>
            <property name="LLEG_ANKLE_R.mode" value="HighGain"/>
            <property name="RLEG_ANKLE_R.angle" value="0.0 "/>
            <property name="LLEG_HIP_Y.mode" value="HighGain"/>
            <property name="RLEG_HIP_P.mode" value="HighGain"/>
            <property name="RARM_WRIST_P.angle" value="0.0 "/>
            <property name="CHEST.mode" value="HighGain"/>
            <property name="RARM_WRIST_R.angle" value="0.0 "/>
            <property name="RARM_WRIST_Y.angle" value="0.0 "/>
            <property name="RLEG_KNEE.angle" value="0.0785047 "/>
            <property name="RLEG_HIP_R.mode" value="HighGain"/>
            <property name="LARM_SHOULDER_P.angle" value="0.174533 "/>
```

Figura 4.3 Ejemplo de archivo XML

Estos archivos se pueden encontrar en la ubicación:

C:\Archivos de programa\OpenHRP-3.0.2\client\gui\project, y dentro de esa ubicación se busca el archivo Nombre_del_proyecto.xml.

Como se muestra en la Figura 4.3, en este archivo se especifican los parámetros de simulación como la base de tiempo, el tiempo de duración de la simulación, etc. Además, también se especifica qué objetos VRML van a ser cargados en el proyecto, por ejemplo el modelo del robot, suelo, cajas, etc.

En todos ellos se puede indicar la posición inicial del objeto en el visor del simulador. En el modelo del robot se pueden definir también los valores iniciales de cada una de las articulaciones que componen el modelo.

También se puede definir el par de colisión entre dos objetos que van a ser cargados en el proyecto. Se pueden especificar las constantes de rozamiento tanto por deslizamiento como estático.

Por último en este archivo se puede crear una lista de gráficas para el proyecto.

4.2 Cómo cargar el proyecto en el simulador

En primer lugar se debe ir al directorio que contiene la interfaz del simulador, ver Figura 4.4. Este directorio es **C:\Archivos de programa\OpenHRP-3.0.2\bin\dos**. A continuación se debe hacer doble click sobre el archivo **GrxUI.bat**.

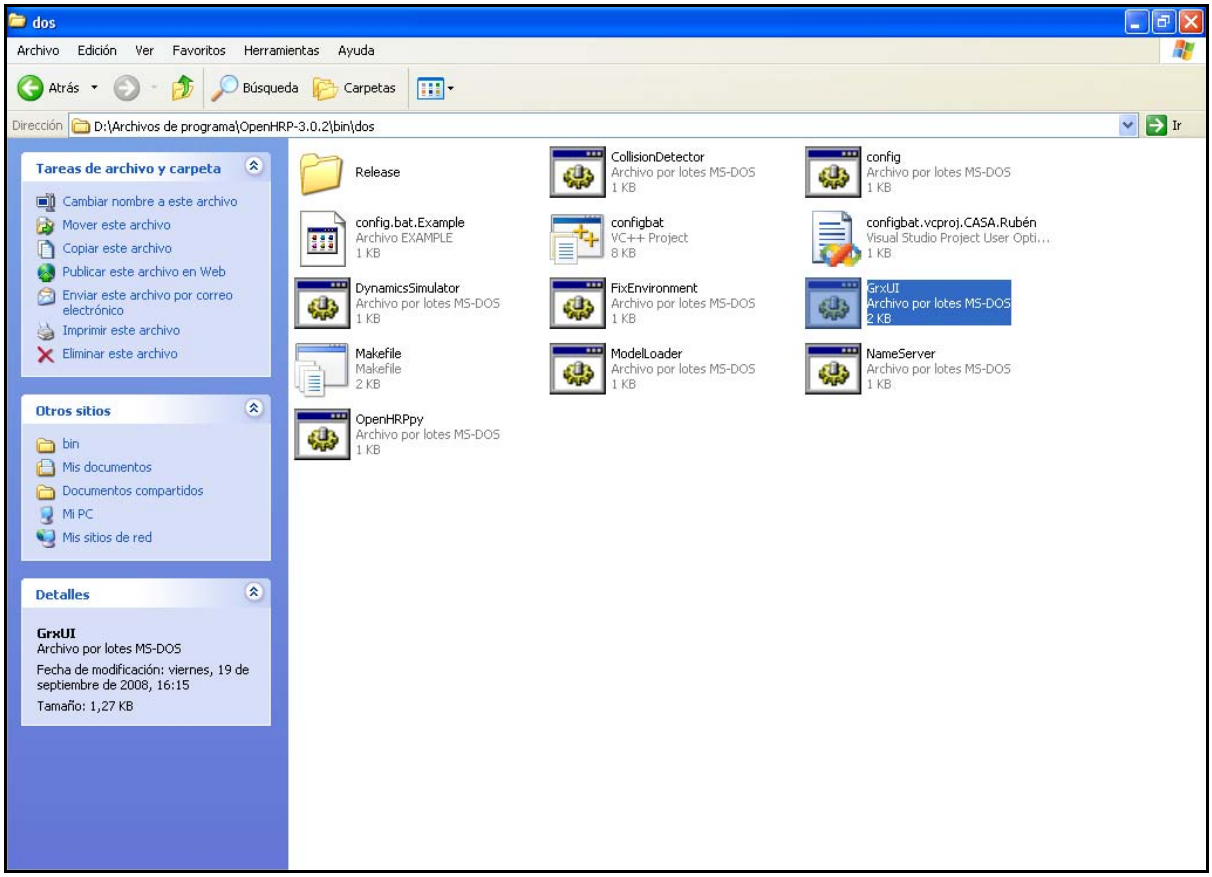


Figura 4.4 Directorio de la interfaz del simulador

Cuando la interfaz del simulador esté abierta se debe cargar el proyecto que se desea simular. Para ello se selecciona el menú File y después Load Project como se muestra en la Figura 4.5.

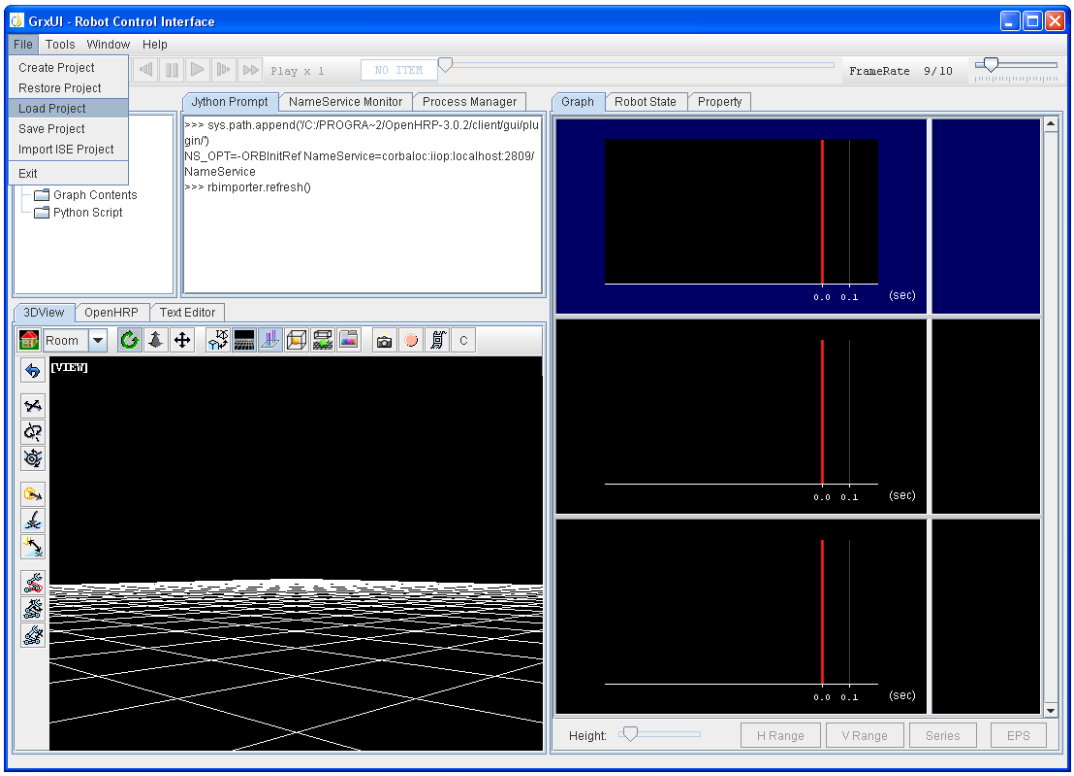


Figura 4.5 Cargar proyecto en el simulador

A continuación, aparece una ventana donde se elige el nombre del proyecto que se va a cargar para la simulación (ver Figura 4.6).

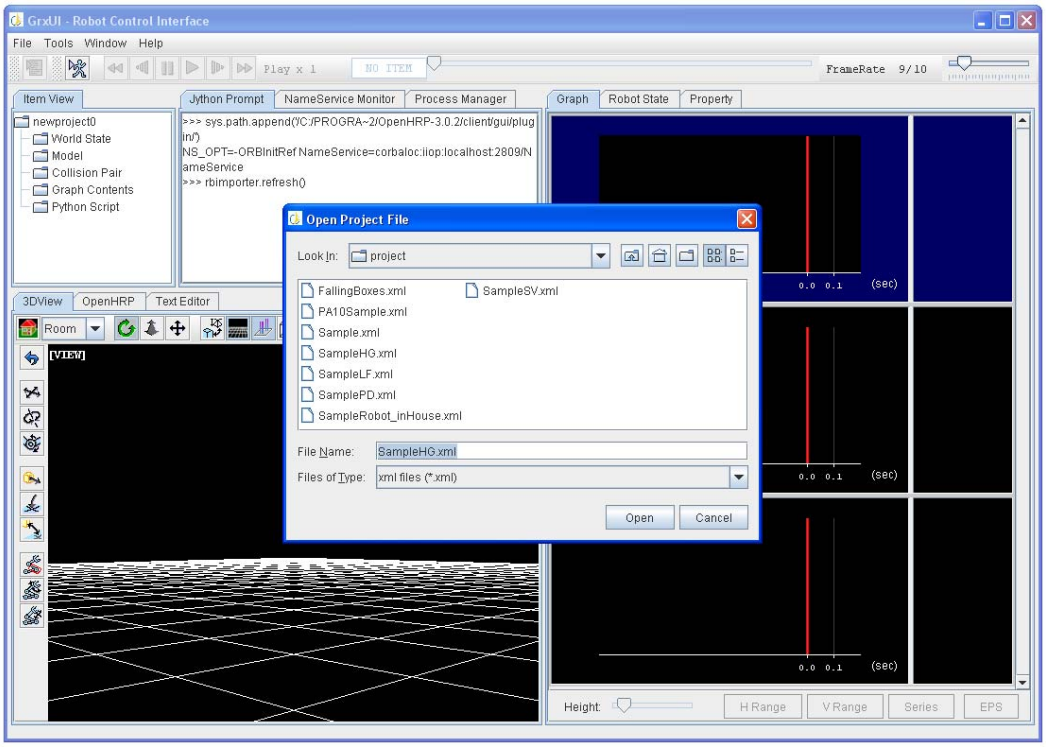


Figura 4.6 Seleccionar proyecto para la simulación

Una vez cargado el proyecto correctamente deben aparecer los objetos que componen dicho proyecto en el simulador, en la parte correspondiente al árbol de módulos (ver Figura 4.7).

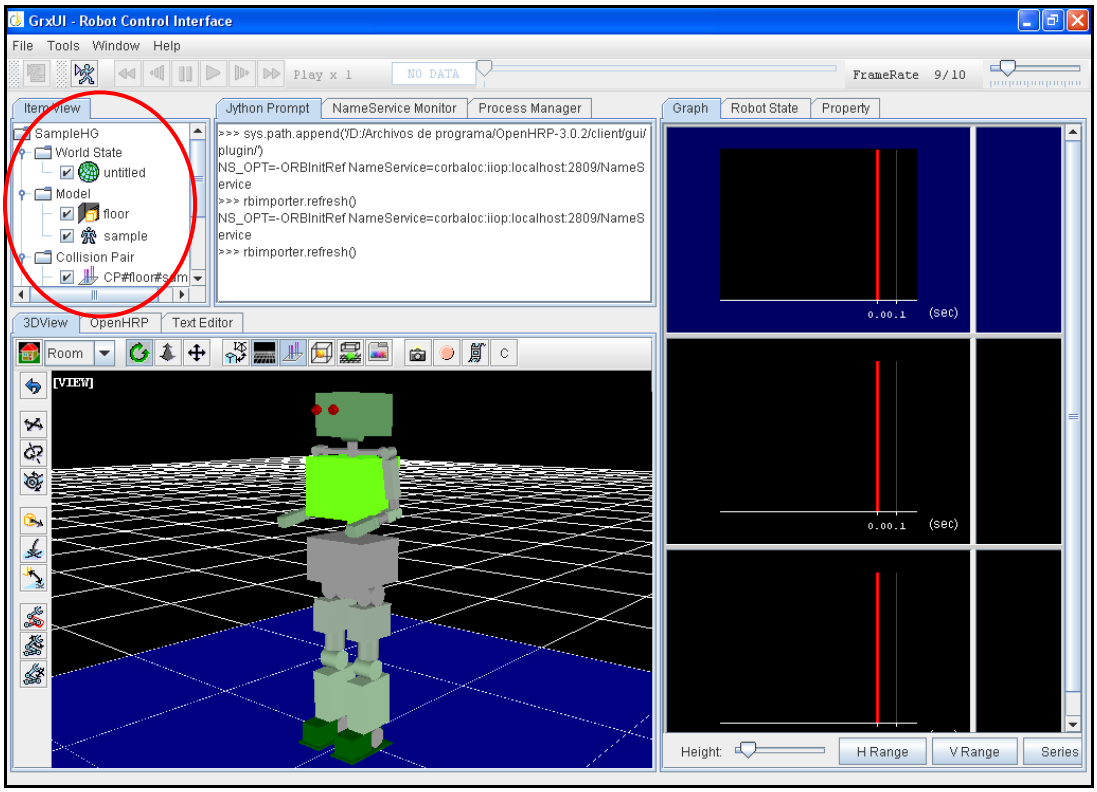


Figura 4.7 Proyecto cargado correctamente

Para comenzar la simulación se debe pulsar el botón "Start Simulation", que viene resaltado en la Figura 4.8

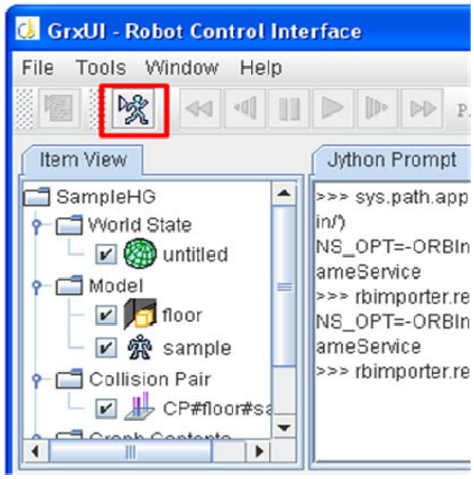


Figura 4.8 Iniciar simulación.

Para suspender o finalizar la simulación se debe pulsar el botón "Suspend Simulation", y a continuación se pulsa "OK" para finalizar la simulación o "Cancel" para continuar con ella (ver Figura 4.9).

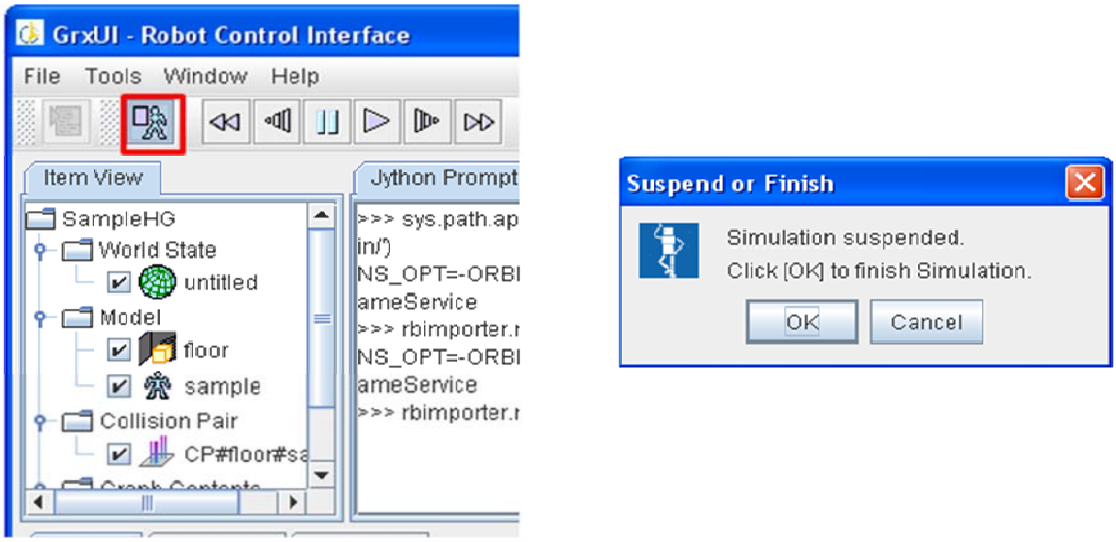


Figura 4.9 Finalizar o suspender simulación.

Para observar los datos de los sensores introducidos en el modelo VRML en las gráficas, se debe seleccionar el botón "Series" que se encuentra en la parte inferior derecha de la interfaz, como muestra la Figura 4.10 [2].

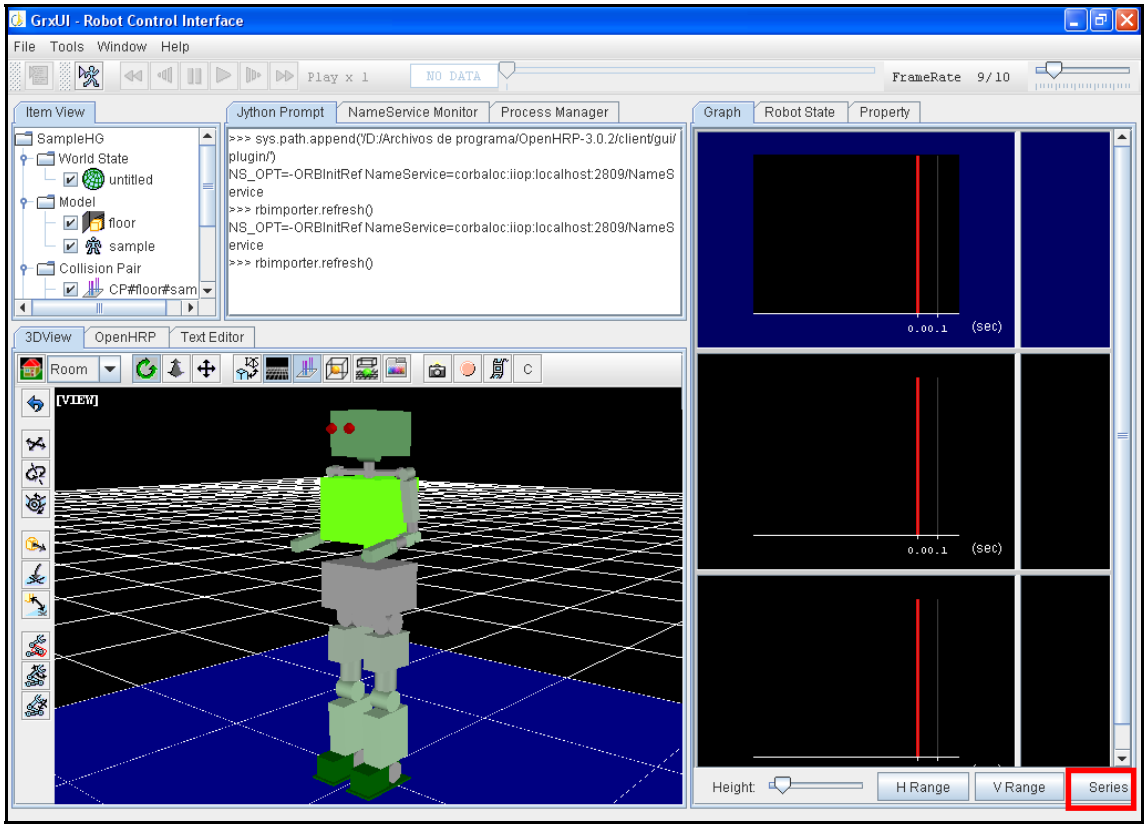


Figura 4.10 Insertar gráfica de sensores

Después de pulsar el botón "Series" aparece un cuadro de diálogo (Figura 4.11) donde se elige el tipo de sensor, el nombre del sensor que se incluye en el modelo que se quiere observar y el atributo. Una vez realizado esto se debe pulsar el botón "Set" y a continuación el botón "Ok".

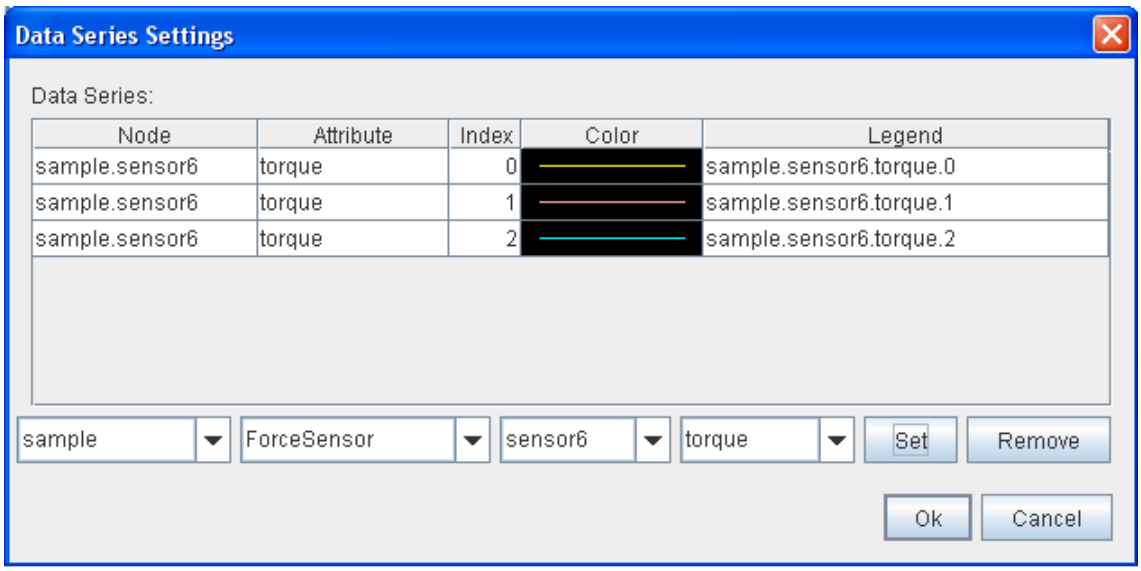


Figura 4.11 Configuración parámetros de la gráfica

Otra posibilidad que nos ofrece OpenHRP3 es guardar en un archivo .csv [2] todos los datos obtenidos de la simulación, como las traslaciones y rotaciones de las articulaciones y también todos los datos que se obtienen de los sensores que se incluyen en el modelo. Para ello, una vez realizada la simulación se selecciona el elemento del árbol que hace referencia a World State que en este caso recibe el nombre de untitled (ver Figura 4.12).

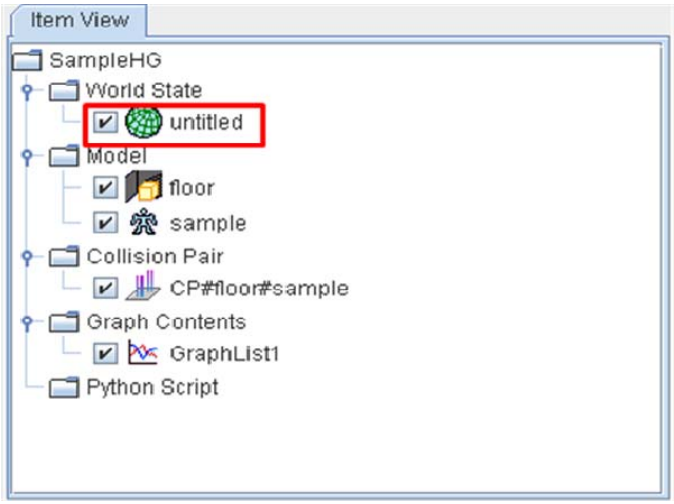


Figura 4.12 Elemento untitled para generación de archivo .csv

Una vez seleccionado, se hace click sobre el elemento con el botón derecho del ratón y se selecciona "saveAsCSV", como se muestra en la Figura 4.13.

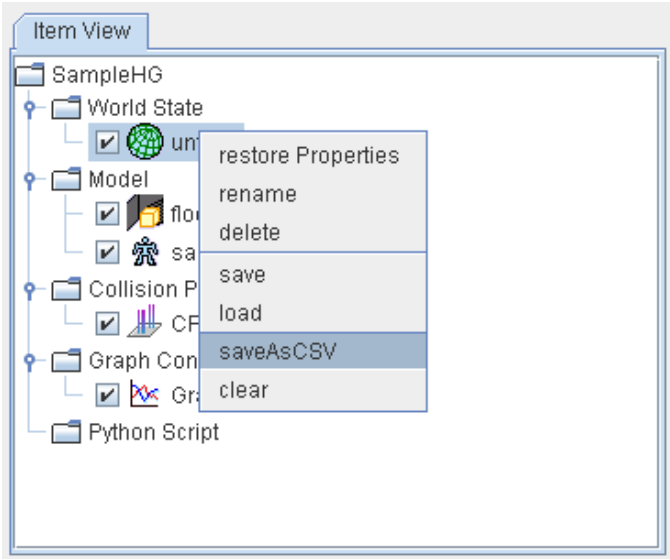


Figura 4.13 Selección saveAsCSV

Como se muestra en la Figura 4.14 este archivo .csv se guarda en el siguiente directorio: C:\Archivos de programa\OpenHRP-3.0.2\client\gui\log.

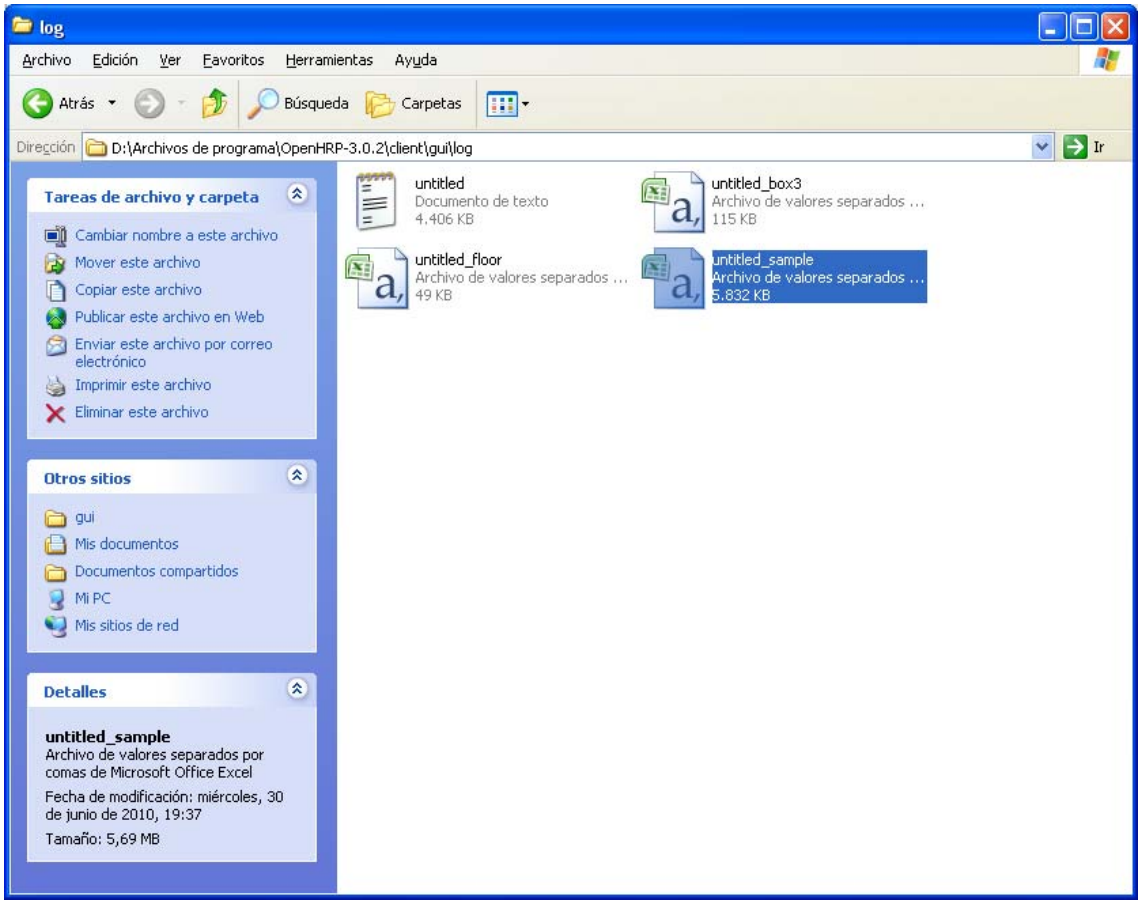


Figura 4.14 Ubicación archivo .csv.

Cabe destacar que el archivo .csv tiene un formato de datos separados por comas, por tanto, al abrilo con Microsoft Excel, aparecerán los datos situados en la primera columna de Excel. Para verlo correctamente se debe pinchar en el botón datos -> Texto en columnas y especificar que separe en columnas cada coma del archivo. Por último se debe tener en cuenta que ante una simulación con demasiados sensores el archivo sólo se verá correctamente en la versión 2007 de Microsoft Excel, ya que para la versión anterior (2003) habría demasiadas columnas y no sería posible ordenarlos correctamente.

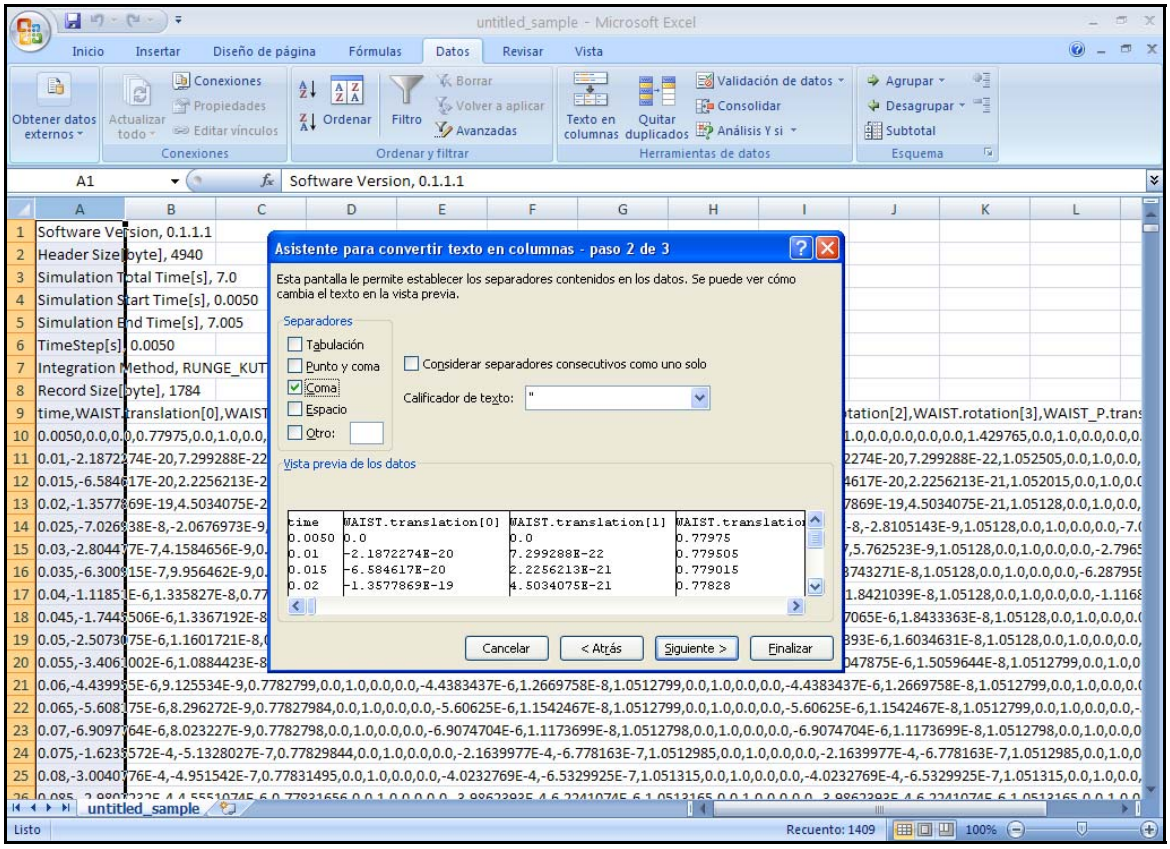


Figura 4.15 Ordenación de los datos del archivo .csv.

5 Simulación de tareas del robot Rh-2 en OpenHRP3

5.1 Introducción

Para la validación del modelo en el simulador OpenHRP3 se usarán unos datos de trayectorias que se nos han proporcionado sacados de diferentes simulaciones realizadas en otra plataforma, para intentar una caminada del robot (que consiste en dos pasos), y de las cuales se comentará sus resultados más adelante.

Todos los archivos necesarios para la simulación de estas tareas se encuentran en el CD adjunto a la memoria.

A continuación se explicará brevemente el proceso de introducción de las trayectorias en los archivos .dat explicados en el punto anterior, necesarios para la realización de la simulación.

5.2 Introducción de trayectorias en los archivos .dat de simulación

Como el objetivo es realizar una caminada del robot, los datos de trayectorias que se han proporcionado, son las trayectorias de las piernas del robot.
Estos datos fueron obtenidos en una simulación de 7 segundos con una base de tiempos de 0.005 segundos.

En los datos proporcionados se incluyen las trayectorias en posición y velocidad de las siguientes articulaciones, en las que además se observa el sentido de giro de cada una de ellas:

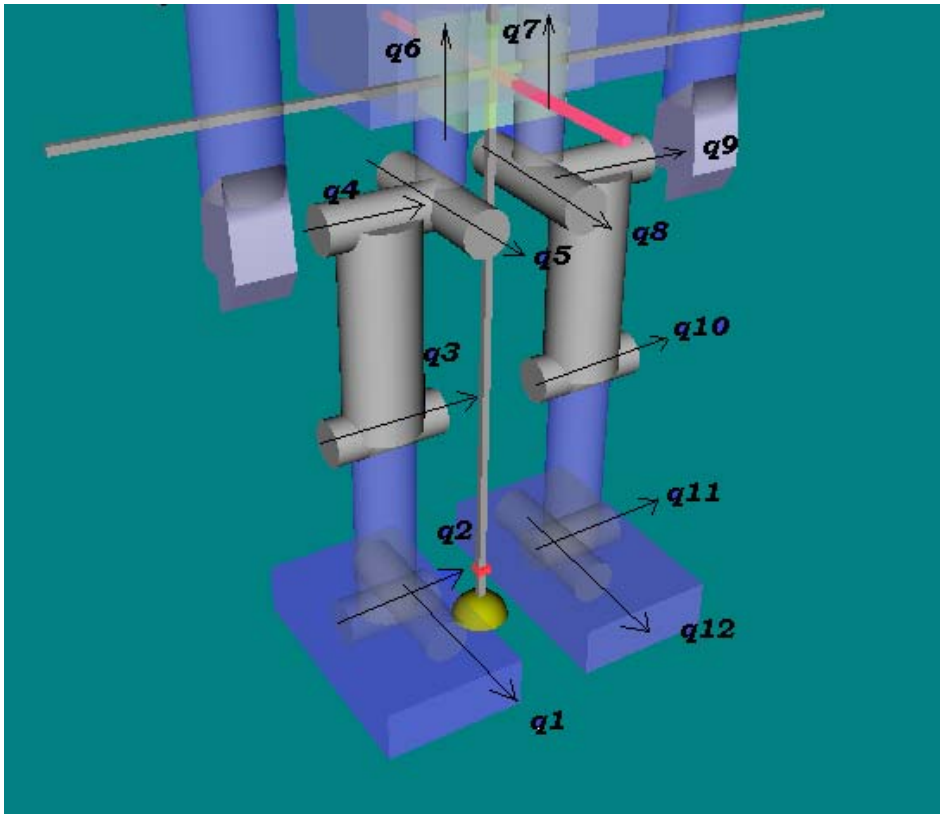


Figura 5.1 Articulaciones de los datos de trayectorias

Por tanto se debe buscar la equivalencia entre estas articulaciones y las definidas mediante el código en VRML para saber dónde se han de colocar los datos en los archivos .dat:

ARTICULACIONES EN LOS DATOS	ARTICULACIONES OPENHRP3
q1	RLEG_ANKLE_R
q2	RLEG_ANKLE_P
q3	RLEG_KNEE
q4	RLEG_HIP_P
q5	RLEG_HIP_R
q6	RLEG_HIP_Y
q7	LLEG_HIP_Y
q8	LLEG_HIP_R
q9	LLEG_HIP_P
q10	LLEG_KNEE
q11	LLEG_ANKLE_P
q12	LLEG_ANKLE_R

Tabla 5.1. Equivalencia entre datos de trayectorias y modelo VRML

Los datos se introducirán respetando la primera columna que es la base de tiempos y siguiendo rigurosamente el orden de joyntId declarados en el código para las columnas posteriores. A continuación se muestra un ejemplo de cómo serían los archivos de velocidad proporcionados y como quedaría en el archivo .dat:

0	0	0	0	0	0	0	0	0	0	0	0
0.005	0	-7.1254E-06	1.4251E-05	-7.1254E-06	0	0	0	0	-7.1254E-06	1.4251E-05	-7.1254E-06
0.01	0	-2.8445E-05	5.6889E-05	-2.8445E-05	0	0	0	0	-2.8445E-05	5.6889E-05	-2.8445E-05
0.015	0	-6.3872E-05	0.00012774	-6.3872E-05	0	0	0	0	-6.3872E-05	0.00012774	-6.3872E-05
0.02	0	-0.00011332	0.00022664	-0.00011332	0	0	0	0	-0.00011332	0.00022664	-0.00011332
0.025	0	-0.00017671	0.00035342	-0.00017671	0	0	0	0	-0.00017671	0.00035342	-0.00017671
0.03	0	-0.00025395	0.0005079	-0.00025395	0	0	0	0	-0.00025395	0.0005079	-0.00025395
0.035	0	-0.00034495	0.00068991	-0.00034495	0	0	0	0	-0.00034495	0.00068991	-0.00034495
0.04	0	-0.00044964	0.00089928	-0.00044964	0	0	0	0	-0.00044964	0.00089928	-0.00044964
0.045	0	-0.00056793	0.00113585	-0.00056793	0	0	0	0	-0.00056793	0.00113585	-0.00056793
0.05	0	-0.00069972	0.00139945	-0.00069972	0	0	0	0	-0.00069972	0.00139945	-0.00069972
0.055	0	-0.00084495	0.0016899	-0.00084495	0	0	0	0	-0.00084495	0.0016899	-0.00084495
0.06	0	-0.00100352	0.00200704	-0.00100352	0	0	0	0	-0.00100352	0.00200704	-0.00100352
0.065	0	-0.00117535	0.0023507	-0.00117535	0	0	0	0	-0.00117535	0.0023507	-0.00117535
0.07	0	-0.00136035	0.0027207	-0.00136035	0	0	0	0	-0.00136035	0.0027207	-0.00136035
0.075	0	-0.00155845	0.00311689	-0.00155845	0	0	0	0	-0.00155845	0.00311689	-0.00155845
0.08	0	-0.00176955	0.0035391	-0.00176955	0	0	0	0	-0.00176955	0.0035391	-0.00176955
0.085	0	-0.00199357	0.00398715	-0.00199357	0	0	0	0	-0.00199357	0.00398715	-0.00199357
0.09	0	-0.00223044	0.00446088	-0.00223044	0	0	0	0	-0.00223044	0.00446088	-0.00223044
0.095	0	-0.00248006	0.00496013	-0.00248006	0	0	0	0	-0.00248006	0.00496013	-0.00248006
0.1	0	-0.00274236	0.00548472	-0.00274236	0	0	0	0	-0.00274236	0.00548472	-0.00274236
0.105	0	-0.00301724	0.00603449	-0.00301724	0	0	0	0	-0.00301724	0.00603449	-0.00301724
0.11	0	-0.00330464	0.00660927	-0.00330464	0	0	0	0	-0.00330464	0.00660927	-0.00330464
0.115	0	-0.00360445	0.00720891	-0.00360445	0	0	0	0	-0.00360445	0.00720891	-0.00360445
0.12	0	-0.00391661	0.00783323	-0.00391661	0	0	0	0	-0.00391661	0.00783323	-0.00391661

Figura 5.2 Archivo proporcionado de posición de eslabones

0	0	0	0	0	0	0	0	0	0	0	0
0.005	0	-7.13E-06	0	1.43E-05	-7.13E-06	0	0	0	0	0	0
0.01	0	-2.84E-05	0	5.69E-05	-2.84E-05	0	0	0	0	0	0
0.015	0	-6.39E-05	0	0.00012774	-6.39E-05	0	0	0	0	0	0
0.02	0	-0.00011332	0	0.00022664	-0.00011332	0	0	0	0	0	0
0.025	0	-0.00017671	0	0.00035342	-0.00017671	0	0	0	0	0	0
0.03	0	-0.00025395	0	0.0005079	-0.00025395	0	0	0	0	0	0
0.035	0	-0.00034495	0	0.00068991	-0.00034495	0	0	0	0	0	0
0.04	0	-0.00044964	0	0.00089928	-0.00044964	0	0	0	0	0	0
0.045	0	-0.00056793	0	0.00113585	-0.00056793	0	0	0	0	0	0
0.05	0	-0.00069972	0	0.00139945	-0.00069972	0	0	0	0	0	0
0.055	0	-0.00084495	0	0.0016899	-0.00084495	0	0	0	0	0	0
0.06	0	-0.00100352	0	0.00200704	-0.00100352	0	0	0	0	0	0
0.065	0	-0.00117535	0	0.0023507	-0.00117535	0	0	0	0	0	0
0.07	0	-0.00136035	0	0.0027207	-0.00136035	0	0	0	0	0	0
0.075	0	-0.00155845	0	0.00311689	-0.00155845	0	0	0	0	0	0
0.08	0	-0.00176955	0	0.0035391	-0.00176955	0	0	0	0	0	0
0.085	0	-0.00199357	0	0.00398715	-0.00199357	0	0	0	0	0	0
0.09	0	-0.00223044	0	0.00446088	-0.00223044	0	0	0	0	0	0
0.095	0	-0.00248006	0	0.00496013	-0.00248006	0	0	0	0	0	0
0.1	0	-0.00274236	0	0.00548472	-0.00274236	0	0	0	0	0	0
0.105	0	-0.00301724	0	0.00603449	-0.00301724	0	0	0	0	0	0
0.11	0	-0.00330464	0	0.00660928	-0.00330464	0	0	0	0	0	0
0.115	0	-0.00360446	0	0.00720891	-0.00360446	0	0	0	0	0	0
0.12	0	-0.00391661	0	0.00783323	-0.00391661	0	0	0	0	0	0

Figura 5.3 Archivo angle.dat con las trayectorias introducidas

Si tomamos como ejemplo el eslabón q10 (columna 11 del archivo proporcionado de trayectorias), observamos que corresponde con el joint LLEG_KNEE, el cual tiene un jointId = 15, por tanto, la columna q10 deberemos introducirla en la columna número 17 del archivo angle.dat. Este sería el proceso a seguir para el resto de eslabones. Exactamente lo mismo se debería realizar con el archivo de velocidades vel.dat. Cabe destacar que las trayectorias de los brazos en este caso sería 0, ya que no participan en el movimiento que se va a practicar en la simulación.

5.3 Caminada inestable del robot Rh-2

El primer caso práctico que se va a desarrollar es una trayectoria de 2 pasos del robot Rh-2 que se realiza de manera inestable, y por tanto, desequilibra y hace caer el robot. El robot comienza con los dos pies apoyados en el suelo, para después iniciar el paso con la pierna derecha, y posteriormente realizar el segundo paso con la pierna izquierda.

En la Figura 5.4 y Figura 5.5 se pueden ver una parte de los archivos angle.dat y vel.dat que definen la trayectoria de estos dos pasos que hacen desequilibrarse al robot.

0	0	0	0	0	0	0	0	0	0	0	0
0.005	0	-6.85E-06	0	1.44E-05	-7.53E-06	0	0	0	0	0	0
0.01	0	-2.73E-05	0	5.74E-05	-3.01E-05	0	0	0	0	0	0
0.015	0	-6.14E-05	0	0.0001289	-6.75E-05	0	0	0	0	0	0
0.02	0	-0.0001089	0	0.00022869	-0.00011979	0	0	0	0	0	0
0.025	0	-0.00016982	0	0.00035662	-0.0001868	0	0	0	0	0	0
0.03	0	-0.00024404	0	0.00051249	-0.00026845	0	0	0	0	0	0
0.035	0	-0.0003315	0	0.00069615	-0.00036465	0	0	0	0	0	0
0.04	0	-0.00043211	0	0.00090742	-0.00047532	0	0	0	0	0	0
0.045	0	-0.00054578	0	0.00114613	-0.00060036	0	0	0	0	0	0
0.05	0	-0.00067244	0	0.00141211	-0.00073968	0	0	0	0	0	0
0.055	0	-0.000812	0	0.00170519	-0.0008932	0	0	0	0	0	0
0.06	0	-0.00096438	0	0.0020252	-0.00106082	0	0	0	0	0	0
0.065	0	-0.00112951	0	0.00237197	-0.00124246	0	0	0	0	0	0
0.07	0	-0.0013073	0	0.00274533	-0.00143803	0	0	0	0	0	0
0.075	0	-0.00149767	0	0.0031451	-0.00164743	0	0	0	0	0	0
0.08	0	-0.00170054	0	0.00357113	-0.00187059	0	0	0	0	0	0
0.085	0	-0.00191583	0	0.00402323	-0.00210741	0	0	0	0	0	0
0.09	0	-0.00214345	0	0.00450125	-0.0023578	0	0	0	0	0	0
0.095	0	-0.00238334	0	0.00500501	-0.00262167	0	0	0	0	0	0
0.1	0	-0.00263541	0	0.00553435	-0.00289895	0	0	0	0	0	0
0.105	0	-0.00289957	0	0.0060891	-0.00318953	0	0	0	0	0	0
0.11	0	-0.00317576	0	0.00666909	-0.00349333	0	0	0	0	0	0
0.115	0	-0.00346388	0	0.00727415	-0.00381027	0	0	0	0	0	0
0.12	0	-0.00376386	0	0.00790412	-0.00414025	0	0	0	0	0	0

Figura 5.4 Archivo angle.dat trayectoria inestable

0	0	-0.00136951	0	0.00287596	-0.00150646	0	0	0	0	0	0
0.005	0	-0.00409754	0	0.00860482	-0.00450729	0	0	0	0	0	0
0.01	0	-0.0068091	0	0.01429912	-0.00749001	0	0	0	0	0	0
0.015	0	-0.00950425	0	0.01995892	-0.01045467	0	0	0	0	0	0
0.02	0	-0.01218298	0	0.02558427	-0.01340128	0	0	0	0	0	0
0.025	0	-0.01484534	0	0.03117522	-0.01632988	0	0	0	0	0	0
0.03	0	-0.01749135	0	0.03673183	-0.01924048	0	0	0	0	0	0
0.035	0	-0.02012103	0	0.04225416	-0.02213313	0	0	0	0	0	0
0.04	0	-0.02273441	0	0.04774226	-0.02500785	0	0	0	0	0	0
0.045	0	-0.02533151	0	0.05319618	-0.02786466	0	0	0	0	0	0
0.05	0	-0.02791237	0	0.05861598	-0.03070361	0	0	0	0	0	0
0.055	0	-0.030477	0	0.06400171	-0.03352471	0	0	0	0	0	0
0.06	0	-0.03302544	0	0.06935343	-0.03632799	0	0	0	0	0	0
0.065	0	-0.03555771	0	0.07467119	-0.03911348	0	0	0	0	0	0
0.07	0	-0.03807383	0	0.07995506	-0.04188122	0	0	0	0	0	0
0.075	0	-0.04057384	0	0.08520507	-0.04463124	0	0	0	0	0	0
0.08	0	-0.04305776	0	0.0904213	-0.04736355	0	0	0	0	0	0
0.085	0	-0.04552561	0	0.0956038	-0.05007819	0	0	0	0	0	0
0.09	0	-0.04797742	0	0.10075262	-0.05277519	0	0	0	0	0	0
0.095	0	-0.05041323	0	0.10586781	-0.05545458	0	0	0	0	0	0
0.1	0	-0.05283304	0	0.11094944	-0.0581164	0	0	0	0	0	0
0.105	0	-0.05523691	0	0.11599756	-0.06076065	0	0	0	0	0	0
0.11	0	-0.05762484	0	0.12101223	-0.06338739	0	0	0	0	0	0
0.115	0	-0.05999686	0	0.1259935	-0.06599664	0	0	0	0	0	0
0.12	0	-0.06235301	0	0.13094143	-0.06858842	0	0	0	0	0	0

Figura 5.5 Archivo vel.dat trayectoria inestable

Para poder apreciar de forma más precisa lo que ocurre en la simulación, se muestran como en el caso anterior, las gráficas donde se puede observar la posición de cada una de las articulaciones importantes a la hora de realizar los pasos (caderas, rodillas y tobillos). Se divide en sub-apartados para facilitar la comprensión de tal manera que se vea el resultado en la pierna derecha, la pierna izquierda, y las imágenes del simulador:

5.3.1 Pierna derecha

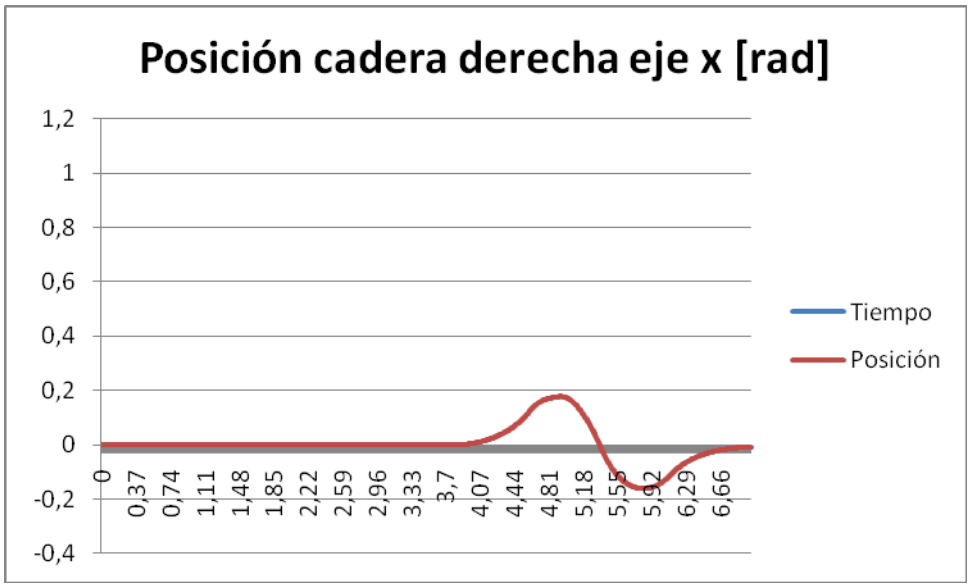


Figura 5.6 Gráfica posiciones de la cadera derecha a lo largo de la simulación (eje x)

En la Figura 5.6 se puede observar la posición de la cadera derecha en el eje x a lo largo de la simulación. A partir del segundo 4 aproximadamente comienza su movimiento, desplazándose en primer lugar hacia la izquierda hasta alcanzar un valor aproximado de 0.2 rad, y después se mueve hacia la derecha hasta -0.2 rad, para finalmente volver a su posición inicial. El tramo comprendido entre 4 y 6.5 segundos es el tiempo en el que transcurren los dos pasos del robot.

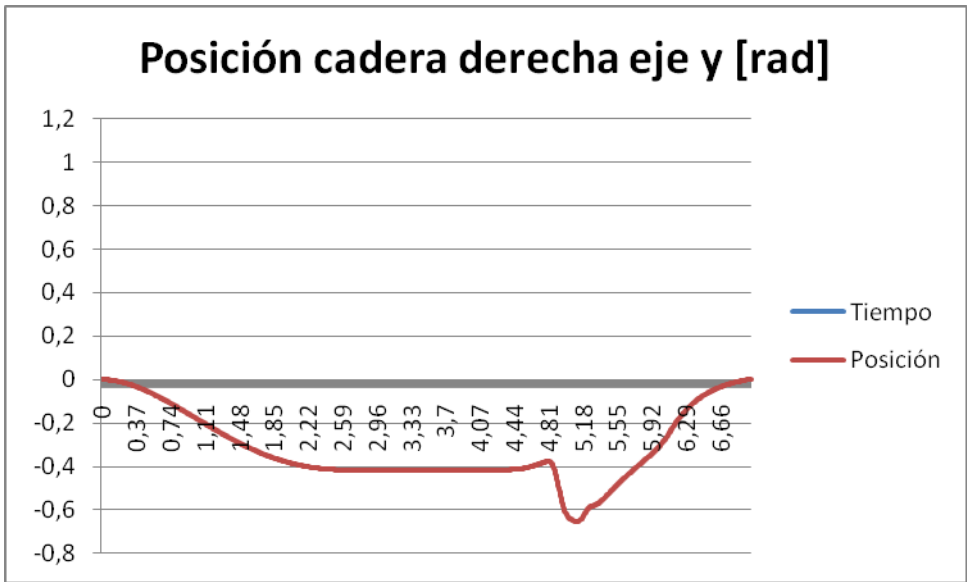


Figura 5.7 Gráfica posiciones de la cadera derecha a lo largo de la simulación (eje y)

La Figura 5.7 muestra la posición de la cadera derecha en el eje y a lo largo de la simulación. Ésta comienza su movimiento desde el principio de la simulación, desplazándose durante los dos primeros segundos hacia delante, en el inicio del movimiento del paso. Se aprecia una variación brusca del valor de la posición entre los instantes 4.81 y 5.18 segundos, que son los correspondientes al paso hacia delante de la pierna derecha del robot, donde alcanza una posición máxima de aproximadamente -0.65 rad. En la recta final recupera la posición inicial.

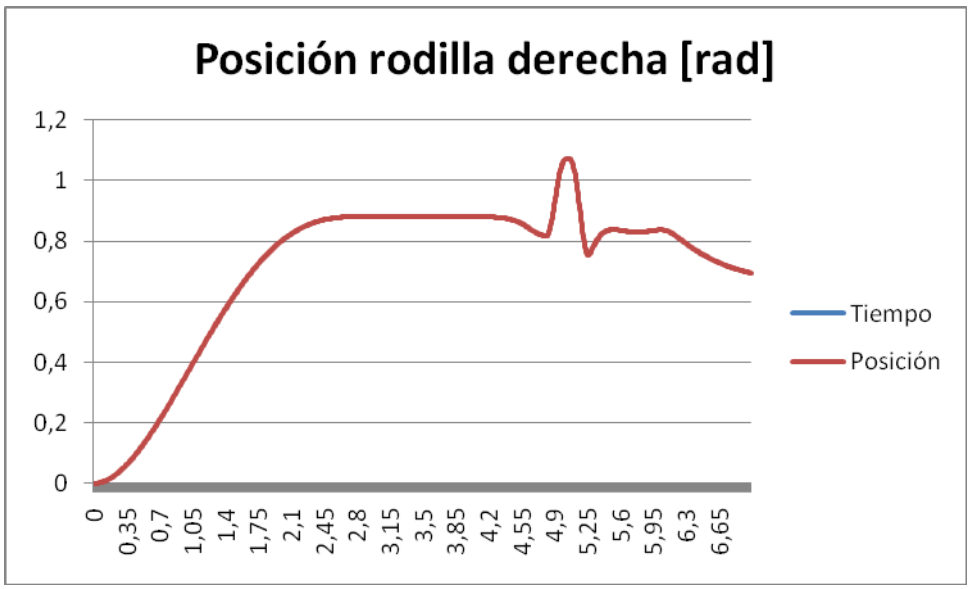


Figura 5.8 Gráfica posiciones de la rodilla derecha a lo largo de la simulación

En 5.8 se obtiene la posición de la rodilla derecha a lo largo de la simulación. Ésta comienza su movimiento desde el principio de la simulación, desplazándose durante los primeros 2.5 segundos hacia atrás en lo que correspondería con el movimiento de flexión de las piernas, justo antes de empezar a dar los pasos. Permanece en esa posición hasta el segundo 4.8 en el que se aprecia una variación brusca del valor de la posición hasta el instante de tiempo 5.25 segundos, lo que correspondería al giro de la rodilla durante el paso hacia delante de la pierna derecha del robot, donde alcanza una posición máxima de aproximadamente 1.1 rad. Posteriormente disminuye hasta la posición de 0.7 rad.

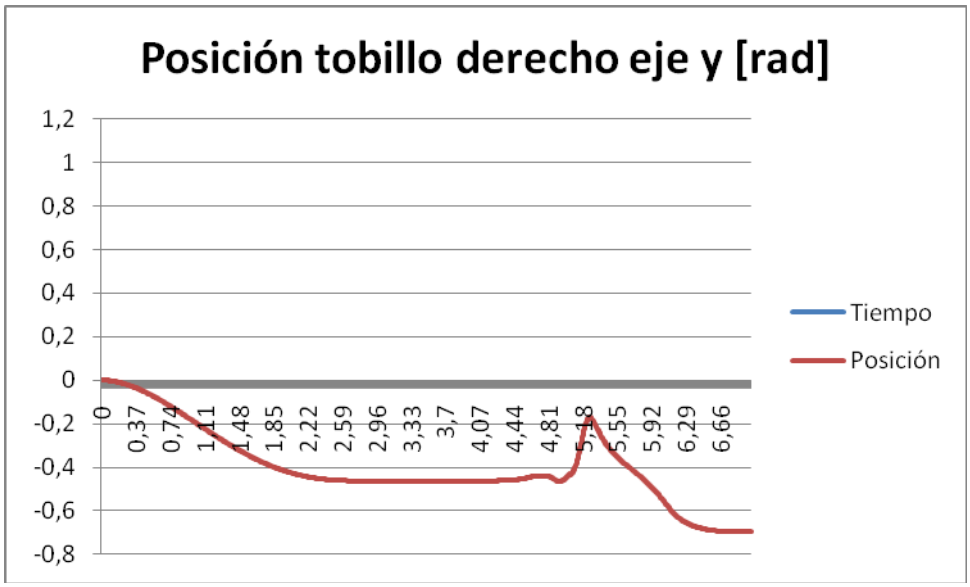


Figura 5.9 Gráfica posiciones del tobillo derecho a lo largo de la simulación (eje y)

En la Figura 5.9 se puede observar la posición del tobillo derecho en el eje y a lo largo de la simulación. Se desplaza desde el principio de la simulación durante los primeros 2.5 segundos hacia delante en lo que correspondería con el movimiento de flexión de las piernas, justo antes de empezar a dar los pasos. Permanece en esa posición hasta el segundo 4.8 en el que se aprecia una variación brusca del valor de la posición hasta el instante de tiempo 5.25 segundos, lo que correspondería al momento en el que el robot apoya la pierna derecha en el suelo, donde alcanza una posición de aproximadamente -0.19 rad. Posteriormente disminuye constantemente hasta el final de la simulación hasta la posición de -0.7 rad.

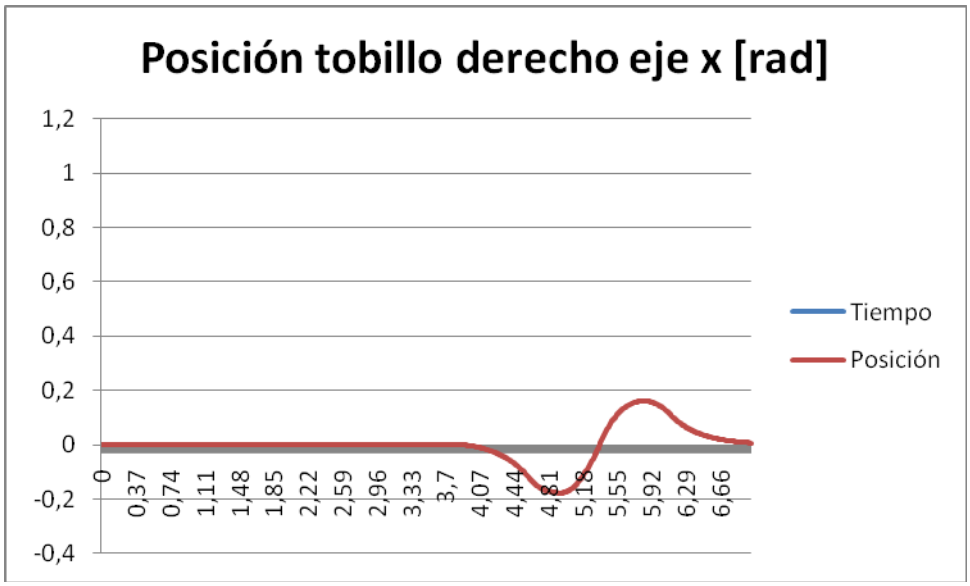


Figura 5.10 Gráfica posiciones del tobillo derecho a lo largo de la simulación (eje x)

La Figura 5.10 muestra la posición del tobillo derecho en el eje x a lo largo de la simulación. Esta articulación comienza su movimiento en el instante de 4 segundos de la simulación, desplazándose durante 1 segundo hacia la derecha y durante el segundo siguiente hacia la izquierda para posteriormente volver a la posición de 0 rad, en lo que correspondería con el movimiento de balanceo de las pierna durante el paso. Los valores que alcanza durante este movimiento son de -0.18 rad cuando gira a la derecha y de 0.18 rad en el giro a la izquierda.

5.3.2 Pierna izquierda

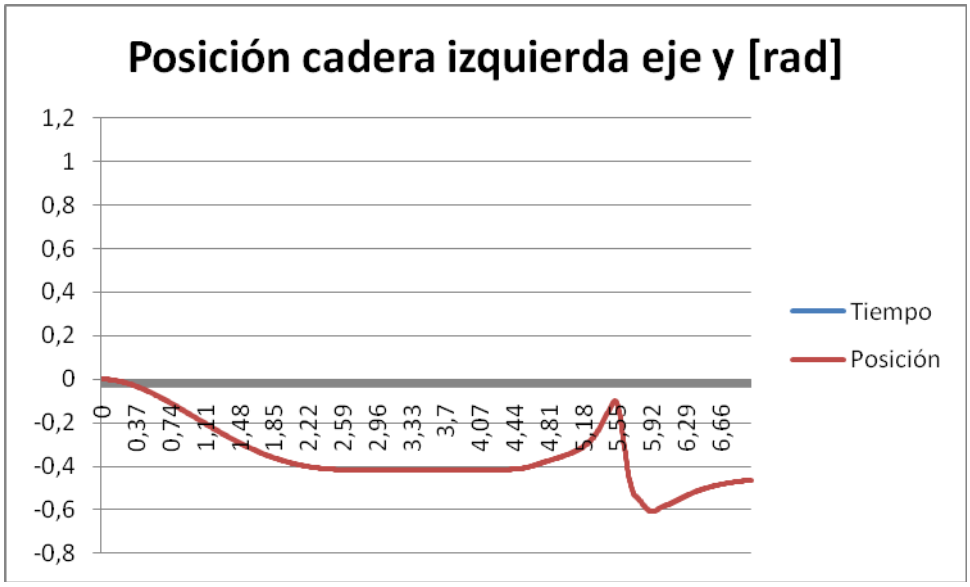


Figura 5.11 Gráfica posiciones de la cadera izquierda a lo largo de la simulación (eje y)

En 5.11 se obtiene la posición de la cadera izquierda en el eje y a lo largo de la simulación. Ésta comienza su movimiento desde el principio de la simulación, desplazándose durante los dos primeros segundos hacia delante, en el inicio del movimiento del paso. Se aprecia una variación brusca del valor de la posición entre los instantes 4.81 y 5.92 segundos, que son los correspondientes al tiempo que tarda el robot en realizar los dos pasos. Se puede comprobar que el inicio del valor de pico comienza durante el paso de la pierna derecha, y éste alcanza su valor mínimo (-0.6 rad) cuando el robot realiza el paso con la pierna izquierda y apoya ésta en el suelo. En la recta final inicia la recuperación de su posición inicial, aunque termina la simulación cuando su posición es de -0.5 rad.

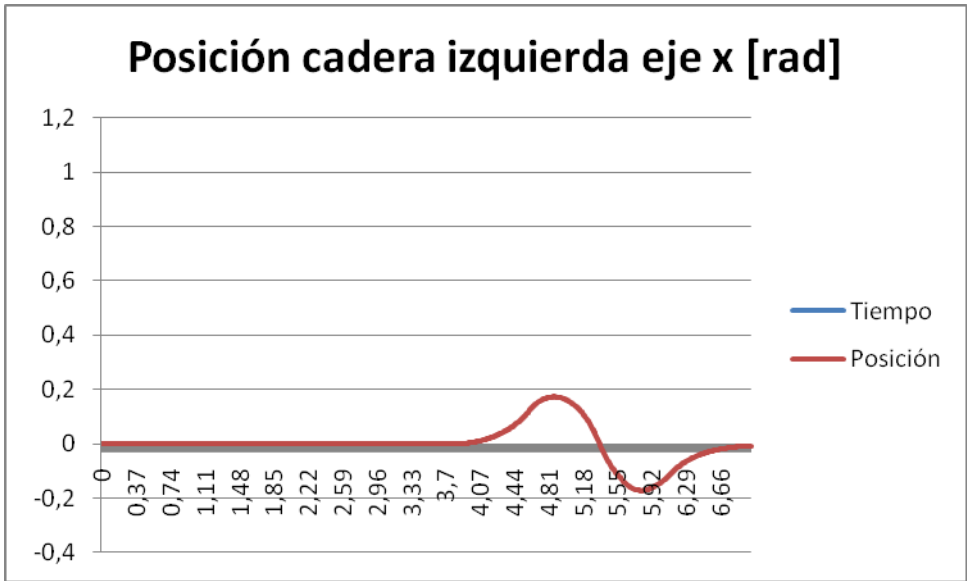


Figura 5.12 Gráfica posiciones de la cadera izquierda a lo largo de la simulación (eje x)

En la Figura 5.12 se puede observar la posición de la cadera izquierda en el eje x a lo largo de la simulación. El movimiento es totalmente análogo al de la cadera derecha en el eje x, por lo que no se explica de nuevo.

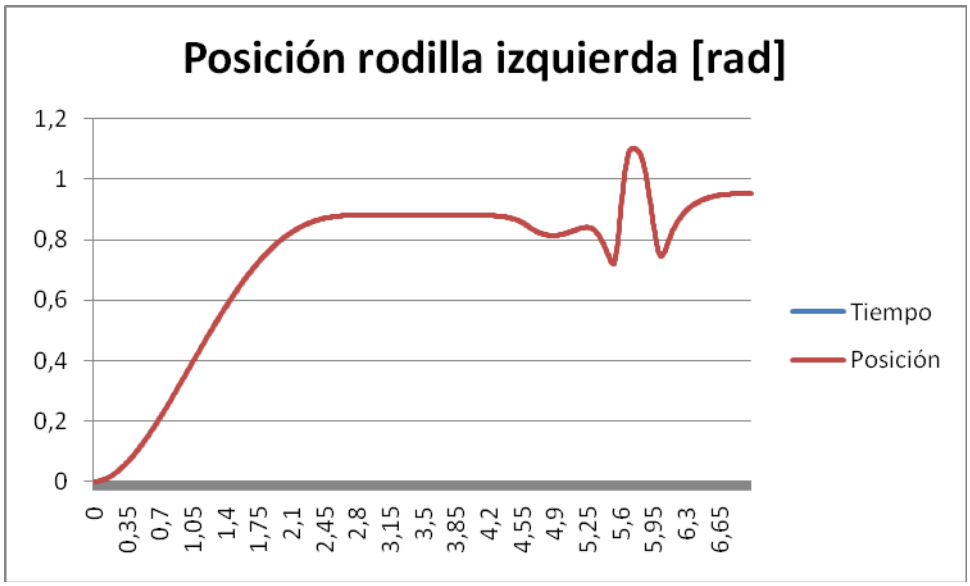


Figura 5.13 Gráfica posiciones de la rodilla izquierda a lo largo de la simulación

La Figura 5.13 muestra la posición de la rodilla izquierda a lo largo de la simulación. Ésta comienza su movimiento desde el principio de la simulación, desplazándose durante los primeros 2.5 segundos hacia atrás en lo que correspondería con el movimiento de flexión de las piernas, justo antes de empezar a dar los pasos. Permanece en esa posición hasta el segundo 4.8 en el que se aprecia una pequeña variación debido al paso de la pierna derecha. Entre los segundos 5.4 y 6 aproximadamente se produce una variación brusca del valor de la posición hasta llegar a alcanzar los 1.1 rad, lo que correspondería al giro de la rodilla durante el paso hacia delante de la pierna izquierda del robot. Posteriormente disminuye hasta la posición de 0.7 rad, para volver a aumentar hasta el final de la simulación llegando hasta los 0.95 rad.

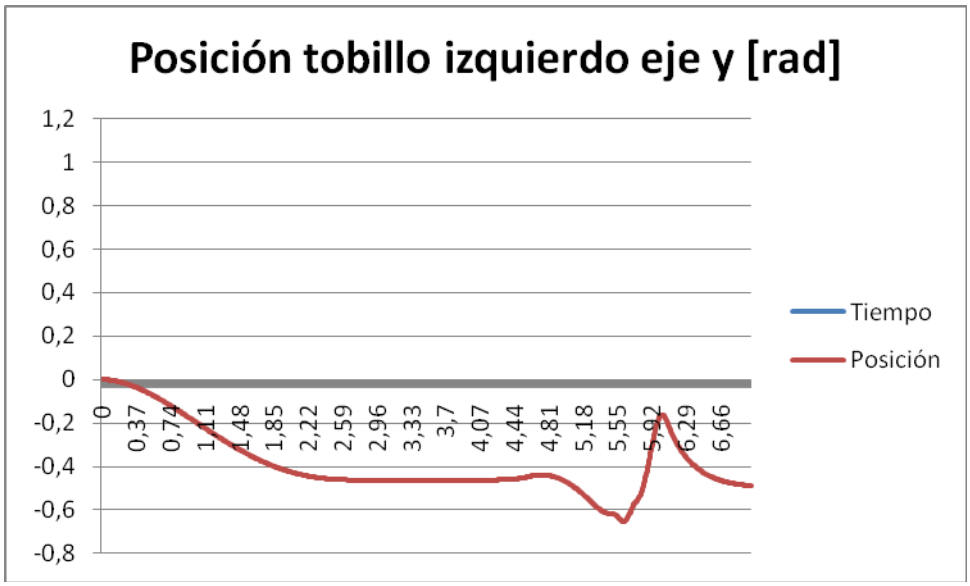


Figura 5.14 Gráfica posiciones del tobillo izquierdo a lo largo de la simulación (eje y)

En 5.14 se obtiene la posición del tobillo izquierdo en el eje y a lo largo de la simulación. Se desplaza desde el principio de la simulación durante los primeros 2.5 segundos hacia delante en lo que correspondería con el movimiento de flexión de las piernas, justo antes de empezar a dar los pasos. Permanece en esa posición hasta el segundo 4.8 en el que se aprecian dos picos de valores provocados por dos variaciones bruscas de posición hasta el instante de tiempo de 6 segundos. El primer pico se produce entre 4.8 y 5.5 segundos y es producido por el paso del robot con la otra pierna (la pierna derecha), llegando a alcanzar la posición de -0.65 rad. El segundo ocurre entre los instantes de tiempo de 5.5 segundos y 6 segundos, y es provocado por el paso con la pierna izquierda, en el que llega a alcanzar el valor aproximado de -0.18 rad. De aquí al final de la simulación disminuye su posición hasta el valor de -0.5 rad.

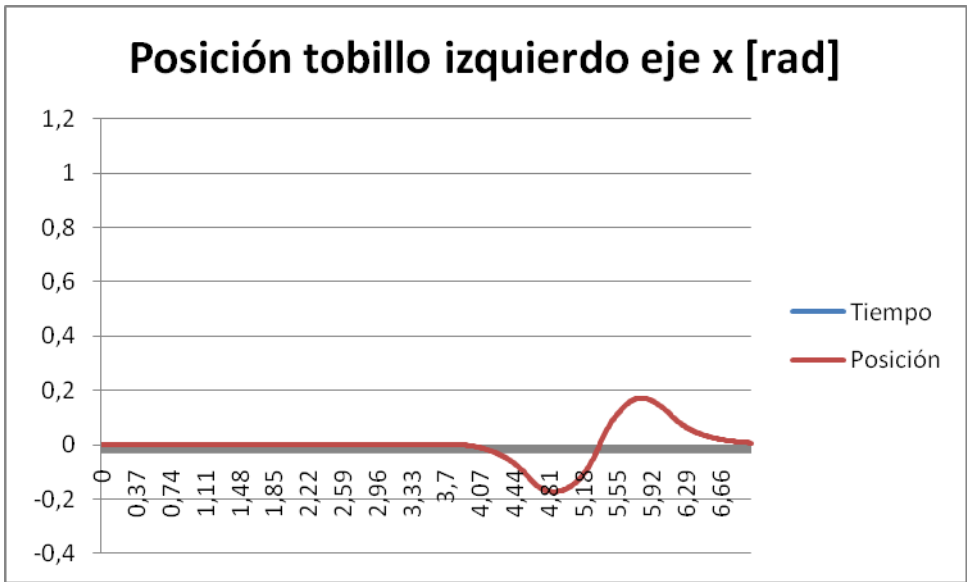


Figura 5.15 Gráfica posiciones del tobillo izquierdo a lo largo de la simulación (eje x)

En la Figura 5.15 se puede observar la posición del tobillo izquierdo en el eje x a lo largo de la simulación. El movimiento es totalmente análogo al del tobillo derecho en el eje x, por lo que no se explica de nuevo.

La velocidad no juega un papel tan crucial como la posición en esta simulación en concreta, por lo que vamos a analizar la gráfica de velocidad sólo de una articulación, en concreto la velocidad de la rodilla izquierda:

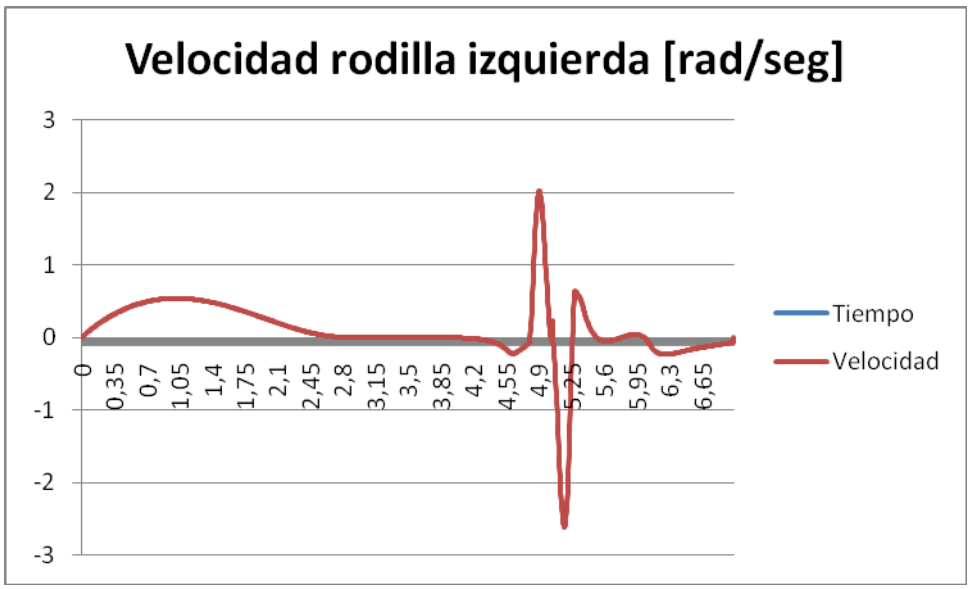


Figura 5.16 Gráfica velocidades de la rodilla izquierda a lo largo de la simulación

Sólo se obtendrá velocidad en las articulaciones cuando haya un cambio de posición en las mismas, lo que se puede observar perfectamente en la Figura 5.16. La rodilla izquierda se mueve al principio realizando el movimiento de flexión de las piernas hasta alcanzar una velocidad de 0.5 rad/seg. El segundo tramo de movimientos se produce entre los segundos 4.55 y 6, donde tienen lugar los dos pasos que da el robot en la simulación. Alcanza unos valores de pico de 2 y -2.5 rad/seg, correspondientes al movimiento de la rodilla primero hacia atrás (sentido anti-horario), y luego hacia delante (sentido horario).

5.3.3 Resultados de la simulación

A continuación se muestran diversas capturas obtenidas en la simulación. Es interesante si se puede, comparar las diferentes figuras con las gráficas mostradas anteriormente, para asociar los diferentes movimientos del robot, con la variación de posiciones. En el CD adjunto a este proyecto se incluye un video de dicha simulación donde se puede comprobar con más certeza los fallos encontrados y el por qué de la inestabilidad.

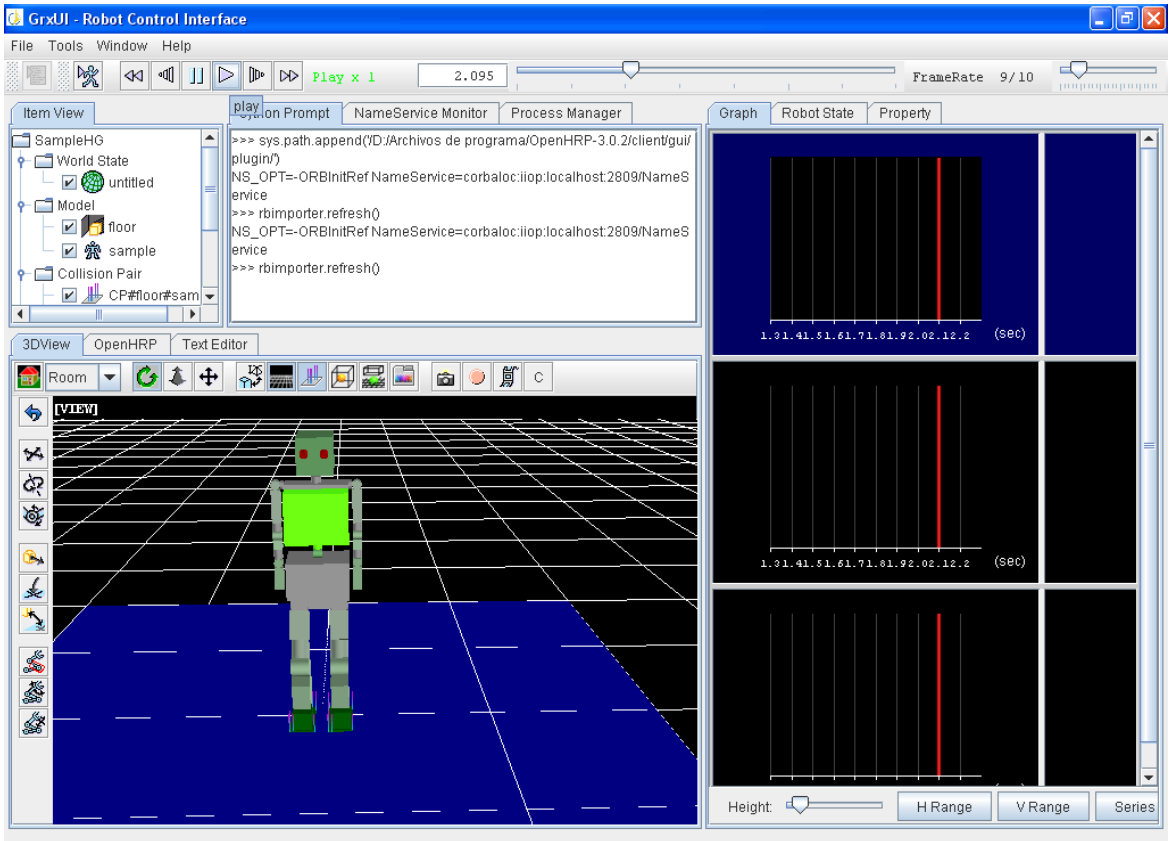


Figura 5.17 Inicio de la simulación. Trayectoria inestable

En este punto el robot se encuentra iniciando la flexión de las piernas para, una vez flexionadas rotar la cadera y comenzar el desplazamiento. Hasta aquí el robot se comporta según lo esperado.

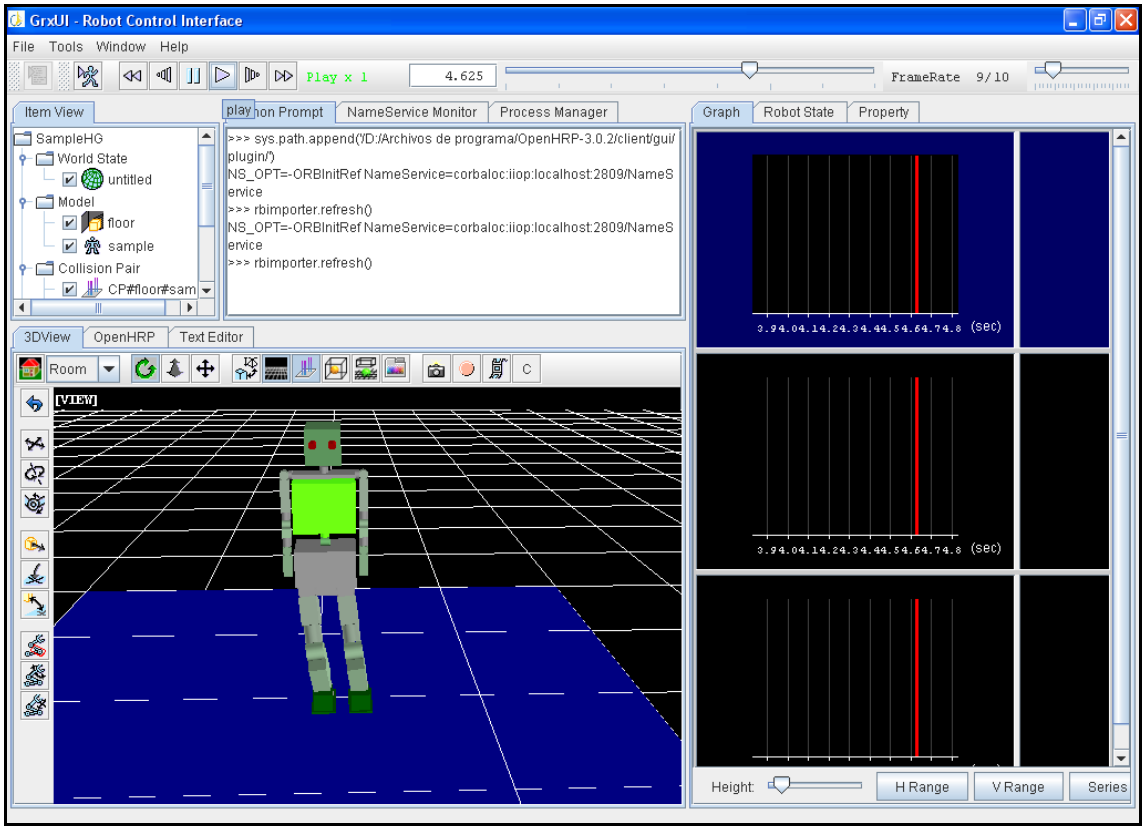


Figura 5.18 Inicio de la rotación de caderas. Trayectoria inestable

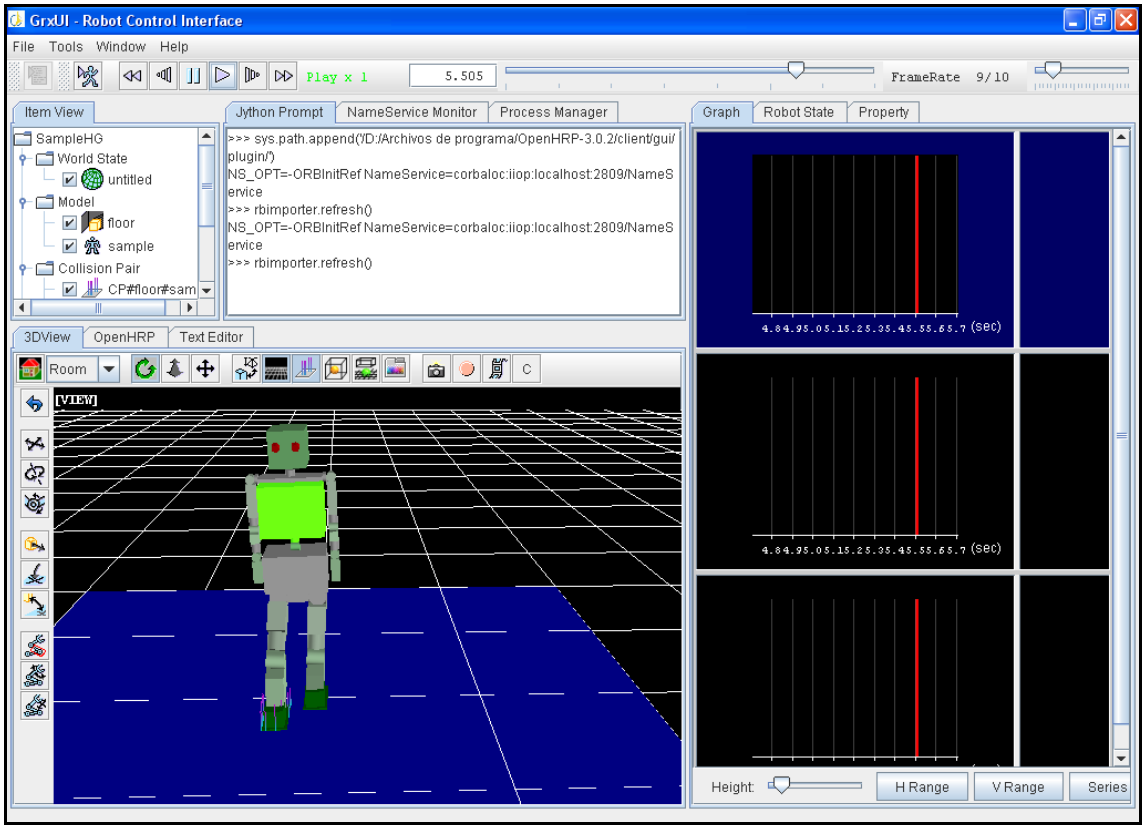


Figura 5.19 Primer paso del robot. Trayectoria inestable

En la Figura 5.18 y la Figura 5.19 se puede observar como el robot a la vez que realiza el movimiento normal de una caminada, tiene un desplazamiento de izquierda a derecha que será una de las causas de su desestabilización. Además se puede observar otro factor determinante en la inestabilidad, el robot da el primer paso con la pierna de apoyo en ese momento (la pierna derecha), lo que le obliga a corregir la posición de la pierna izquierda para realizar ese paso de forma algo más “estable”.

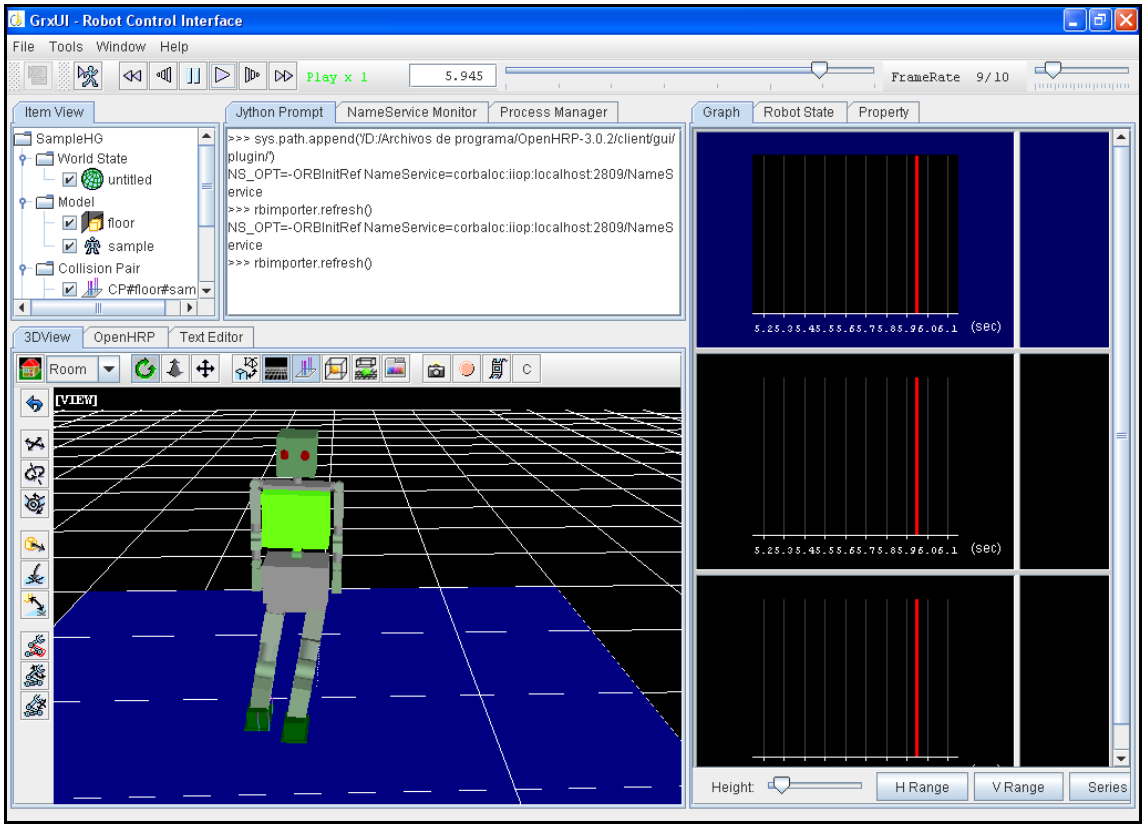


Figura 5.20 Segundo paso del robot. Trayectoria inestable

Se aprecia un nuevo desplazamiento a la vez que hace el movimiento de dar el paso hacia la izquierda. El robot llega ya desestabilizado por el paso anterior.

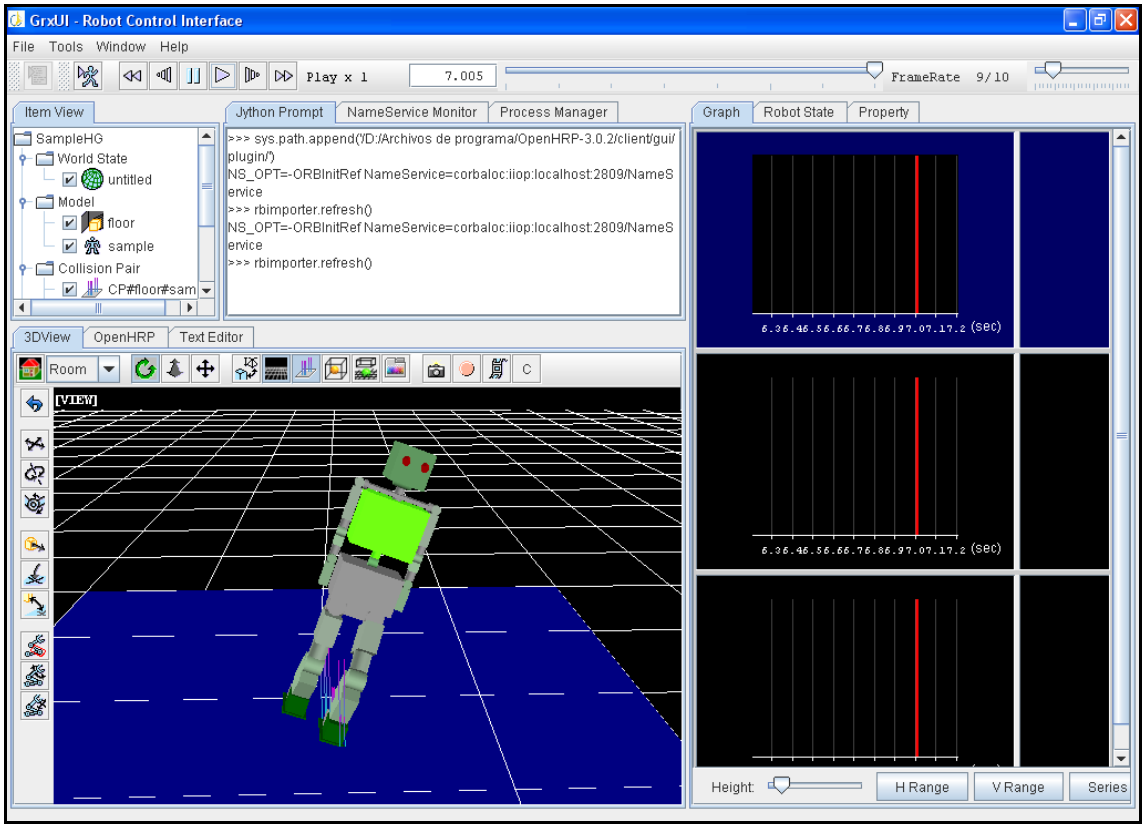


Figura 5.21 Final de la simulación. Trayectoria inestable

Se intuye el comienzo de la caída del robot justo después de dar el segundo paso. Los desplazamientos en las diferentes acciones llevadas a cabo para dar los pasos, y pequeños errores en los valores de las trayectorias, así como el inicio del paso que debería haber sido con la otra pierna (la pierna izquierda), ya que la derecha era la pierna de apoyo en ese momento, han desestabilizados el sistema de control del robot lo suficiente como para hacerle caer. Por tanto se deben ajustar esos valores para que el robot sea capaz de dar el paso con las trayectorias proporcionadas.

5.4 Caminada estable del robot Rh-2

Al igual que en el anterior caso, se va a desarrollar una trayectoria de 2 pasos del robot Rh-2 pero esta vez de manera estable.

El robot comienza con los dos pies apoyados en el suelo, para después iniciar el paso con la pierna derecha, y posteriormente realizar el segundo paso con la pierna izquierda. La principal variación con respecto a los pasos anteriores es el giro de la cadera al inicio del primer paso, que será hacia el otro lado (lado izquierdo), lo que evitará la inestabilidad producida por el pie de apoyo del caso anterior. Esto se podrá ver con más detalle posteriormente.

En la Figura 5.22 y Figura 5.23 se pueden ver una parte de los archivos angle.dat y vel.dat que definen la trayectoria de la caminada estable del robot.

0	0	0	0	0	0	0	0	0	0	0	0
0.005	0	-7.13E-06	0	1.43E-05	-7.13E-06	0	0	0	0	0	0
0.01	0	-2.84E-05	0	5.69E-05	-2.84E-05	0	0	0	0	0	0
0.015	0	-6.39E-05	0	0.00012774	-6.39E-05	0	0	0	0	0	0
0.02	0	-0.00011332	0	0.00022664	-0.00011332	0	0	0	0	0	0
0.025	0	-0.00017671	0	0.00035342	-0.00017671	0	0	0	0	0	0
0.03	0	-0.00025395	0	0.0005079	-0.00025395	0	0	0	0	0	0
0.035	0	-0.00034495	0	0.00068991	-0.00034495	0	0	0	0	0	0
0.04	0	-0.00044964	0	0.00089928	-0.00044964	0	0	0	0	0	0
0.045	0	-0.00056793	0	0.00113585	-0.00056793	0	0	0	0	0	0
0.05	0	-0.00069972	0	0.00139945	-0.00069972	0	0	0	0	0	0
0.055	0	-0.00084495	0	0.0016899	-0.00084495	0	0	0	0	0	0
0.06	0	-0.00100352	0	0.00200704	-0.00100352	0	0	0	0	0	0
0.065	0	-0.00117535	0	0.0023507	-0.00117535	0	0	0	0	0	0
0.07	0	-0.00136035	0	0.0027207	-0.00136035	0	0	0	0	0	0
0.075	0	-0.00155845	0	0.00311689	-0.00155845	0	0	0	0	0	0
0.08	0	-0.00176955	0	0.0035391	-0.00176955	0	0	0	0	0	0
0.085	0	-0.00199357	0	0.00398715	-0.00199357	0	0	0	0	0	0
0.09	0	-0.00223044	0	0.00446088	-0.00223044	0	0	0	0	0	0
0.095	0	-0.00248006	0	0.00496013	-0.00248006	0	0	0	0	0	0
0.1	0	-0.00274236	0	0.00548472	-0.00274236	0	0	0	0	0	0
0.105	0	-0.00301724	0	0.00603449	-0.00301724	0	0	0	0	0	0
0.11	0	-0.00330464	0	0.00660928	-0.00330464	0	0	0	0	0	0
0.115	0	-0.00360446	0	0.00720891	-0.00360446	0	0	0	0	0	0
0.12	0	-0.00391661	0	0.00783323	-0.00391661	0	0	0	0	0	0

Figura 5.22 Archivo angle.dat trayectoria estable

0	0	-0.00142508	0	0.00285017	-0.00142508	0	0	0	0	0	0
0.005	0	-0.00426382	0	0.00852765	-0.00426382	0	0	0	0	0	0
0.01	0	-0.00708544	0	0.01417087	-0.00708544	0	0	0	0	0	0
0.015	0	-0.00988996	0	0.01977991	-0.00988996	0	0	0	0	0	0
0.02	0	-0.0126774	0	0.02535481	-0.0126774	0	0	0	0	0	0
0.025	0	-0.01544781	0	0.03089562	-0.01544781	0	0	0	0	0	0
0.03	0	-0.0182012	0	0.03640239	-0.0182012	0	0	0	0	0	0
0.035	0	-0.0209376	0	0.04187519	-0.0209376	0	0	0	0	0	0
0.04	0	-0.02365703	0	0.04731407	-0.02365703	0	0	0	0	0	0
0.045	0	-0.02635954	0	0.05271907	-0.02635954	0	0	0	0	0	0
0.05	0	-0.02904513	0	0.05809026	-0.02904513	0	0	0	0	0	0
0.055	0	-0.03171384	0	0.06342769	-0.03171384	0	0	0	0	0	0
0.06	0	-0.03436571	0	0.06873141	-0.03436571	0	0	0	0	0	0
0.065	0	-0.03700074	0	0.07400148	-0.03700074	0	0	0	0	0	0
0.07	0	-0.03961898	0	0.07923796	-0.03961898	0	0	0	0	0	0
0.075	0	-0.04222044	0	0.08444089	-0.04222044	0	0	0	0	0	0
0.08	0	-0.04480517	0	0.08961033	-0.04480517	0	0	0	0	0	0
0.085	0	-0.04737317	0	0.09474635	-0.04737317	0	0	0	0	0	0
0.09	0	-0.04992449	0	0.09984898	-0.04992449	0	0	0	0	0	0
0.095	0	-0.05245915	0	0.1049183	-0.05245915	0	0	0	0	0	0
0.1	0	-0.05497718	0	0.10995435	-0.05497718	0	0	0	0	0	0
0.105	0	-0.0574786	0	0.1149572	-0.0574786	0	0	0	0	0	0
0.11	0	-0.05996344	0	0.11992689	-0.05996344	0	0	0	0	0	0
0.115	0	-0.06243174	0	0.12486348	-0.06243174	0	0	0	0	0	0
0.12	0	-0.06488352	0	0.12976704	-0.06488352	0	0	0	0	0	0

Figura 5.23 Archivo vel.dat trayectoria estable

Para poder apreciar de forma más precisa lo que ocurre en la simulación, se muestran como en el caso anterior, las gráficas donde se puede observar la posición de cada una de las articulaciones importantes a la hora de realizar los pasos (caderas, rodillas y tobillos). Al igual que en el apartado anterior se divide en sub-apartados para facilitar la comprensión de tal manera que se vea el resultado en la pierna derecha, la pierna izquierda, y las imágenes del simulador:

5.4.1 Pierna derecha

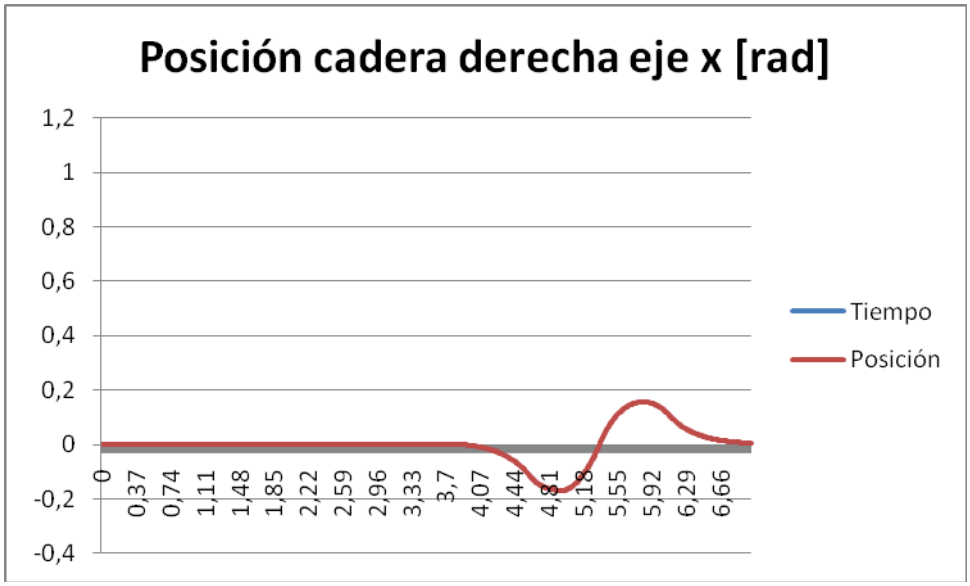


Figura 5.24 Gráfica posiciones de la cadera derecha a lo largo de la simulación (eje x)

En la Figura 5.24 se puede observar la posición de la cadera derecha en el eje x a lo largo de la simulación. A partir del segundo 4 aproximadamente comienza su movimiento, desplazándose en primer lugar hacia la derecha hasta alcanzar un valor aproximado de 0.2 rad, y después se mueve hacia la izquierda hasta -0.2 rad, para finalmente volver a su posición inicial. El tramo comprendido entre 4 y 6.5 segundos es el tiempo en el que transcurren los dos pasos del robot.

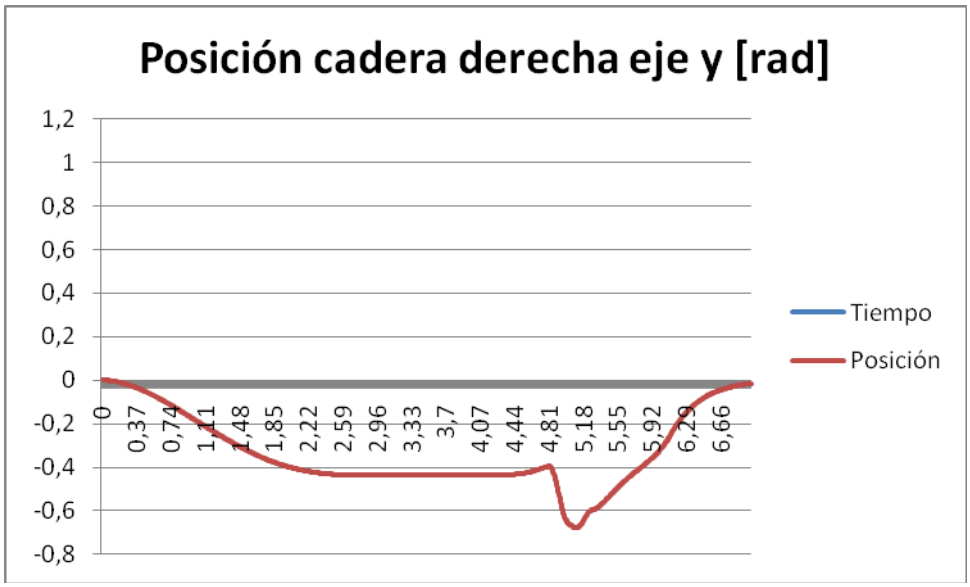


Figura 5.25 Gráfica posiciones de la cadera derecha a lo largo de la simulación (eje y)

La Figura 5.25 muestra la posición de la cadera derecha en el eje y a lo largo de la simulación. Ésta comienza su movimiento desde el principio de la simulación, desplazándose durante los dos primeros segundos hacia delante, en el inicio del movimiento del paso. Se aprecia una variación brusca del valor de la posición entre los instantes 4.81 y 5.18 segundos, que son los correspondientes al paso hacia delante de la pierna derecha del robot, donde alcanza una posición máxima de aproximadamente -0.65 rad. En el tramo final recupera la posición inicial.

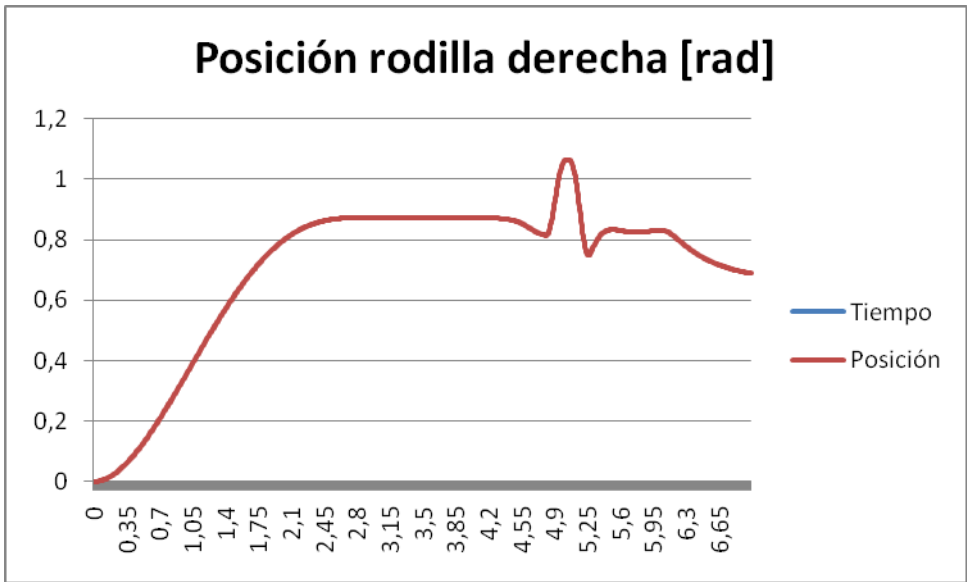


Figura 5.26 Gráfica posiciones de la rodilla derecha a lo largo de la simulación

En 5.26 se obtiene la posición de la rodilla derecha a lo largo de la simulación. Ésta comienza su movimiento desde el principio de la simulación, desplazándose durante los primeros 2.5 segundos hacia atrás en lo que correspondería con el movimiento de flexión de las piernas, justo antes de empezar a dar los pasos. Permanece en esa posición hasta el segundo 4.8 en el que se aprecia una variación brusca del valor de la posición hasta el instante de tiempo de 5.25 segundos, lo que correspondería al giro de la rodilla durante el paso hacia delante de la pierna derecha del robot, donde alcanza una posición máxima de aproximadamente 1.1 rad. Posteriormente disminuye hasta la posición de 0.7 rad.

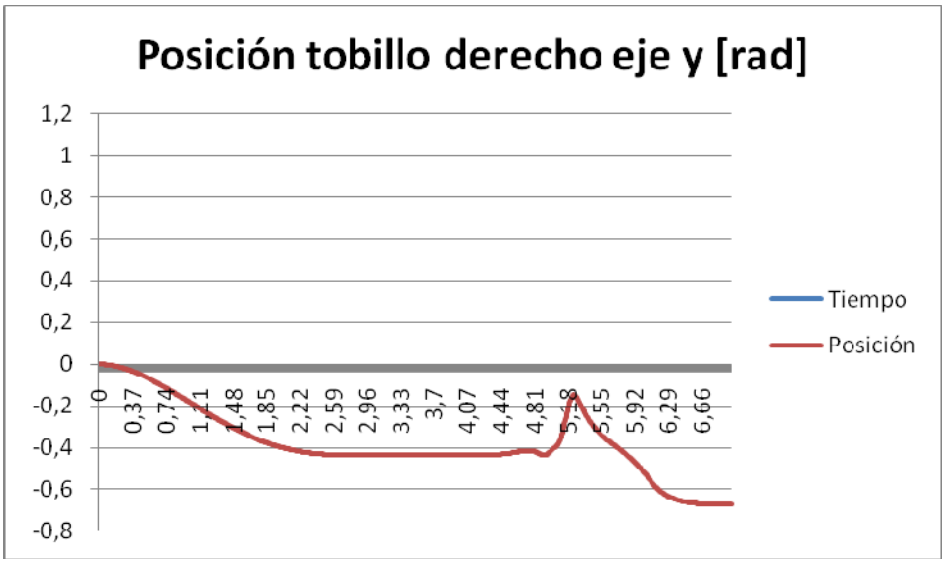


Figura 5.27 Gráfica posiciones del tobillo derecho a lo largo de la simulación (eje y)

En la Figura 5.27 se puede observar la posición del tobillo derecho en el eje y a lo largo de la simulación. Se desplaza desde el principio de la simulación durante los primeros 2.5 segundos hacia delante en lo que correspondería con el movimiento de flexión de las piernas, justo antes de empezar a dar los pasos. Permanece en esa posición hasta el segundo 4.8 en el que se aprecia una variación brusca del valor de la posición hasta el instante de tiempo 5.25 segundos, lo que correspondería al momento en el que el robot apoya la pierna derecha en el suelo, donde alcanza una posición de aproximadamente -0.19 rad. Posteriormente disminuye constantemente hasta el final de la simulación hasta la posición de -0.7 rad.

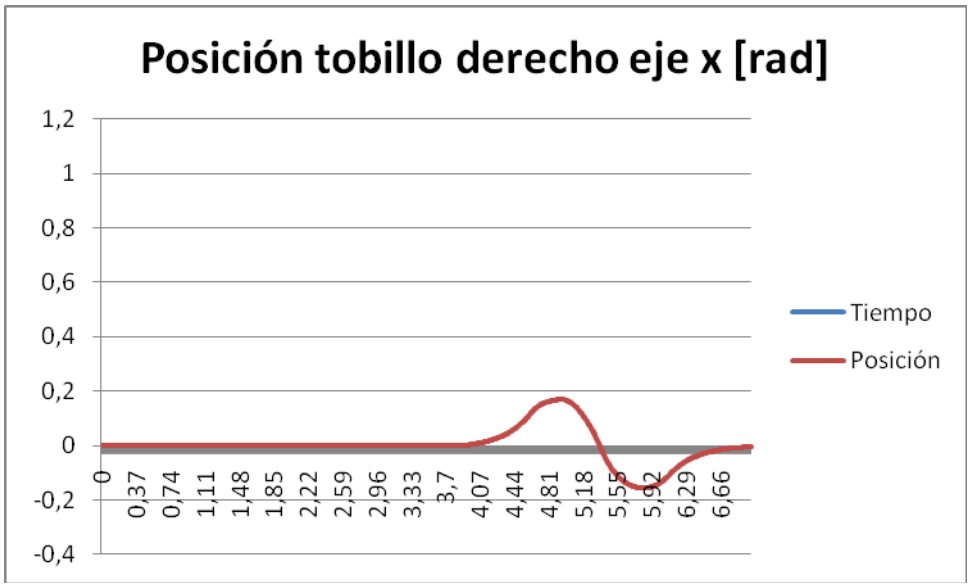


Figura 5.28 Gráfica posiciones del tobillo derecho a lo largo de la simulación (eje x)

La Figura 5.28 muestra la posición del tobillo derecho en el eje x a lo largo de la simulación. Esta articulación comienza su movimiento en el instante de 4 segundos de la simulación, desplazándose durante 1 segundo hacia la izquierda y durante el segundo siguiente hacia la derecha para posteriormente volver a la posición de 0 rad, en lo que correspondería con el movimiento de balanceo de las pierna durante el paso. Los valores que alcanza durante este movimiento son de 0.18 rad cuando gira a la izquierda y de -0.18 rad en el giro a la derecha.

5.4.2 Pierna izquierda

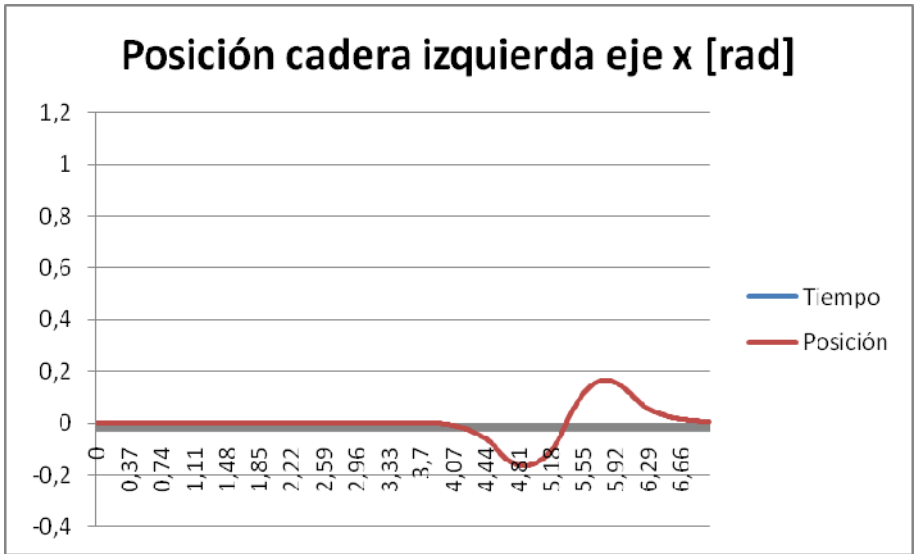


Figura 5.29 Gráfica posiciones de la cadera izquierda a lo largo de la simulación (eje x)

En la Figura 5.29 se puede observar la posición de la cadera izquierda en el eje x a lo largo de la simulación. El movimiento es totalmente análogo al de la cadera derecha en el eje x, por lo que no se explica de nuevo.

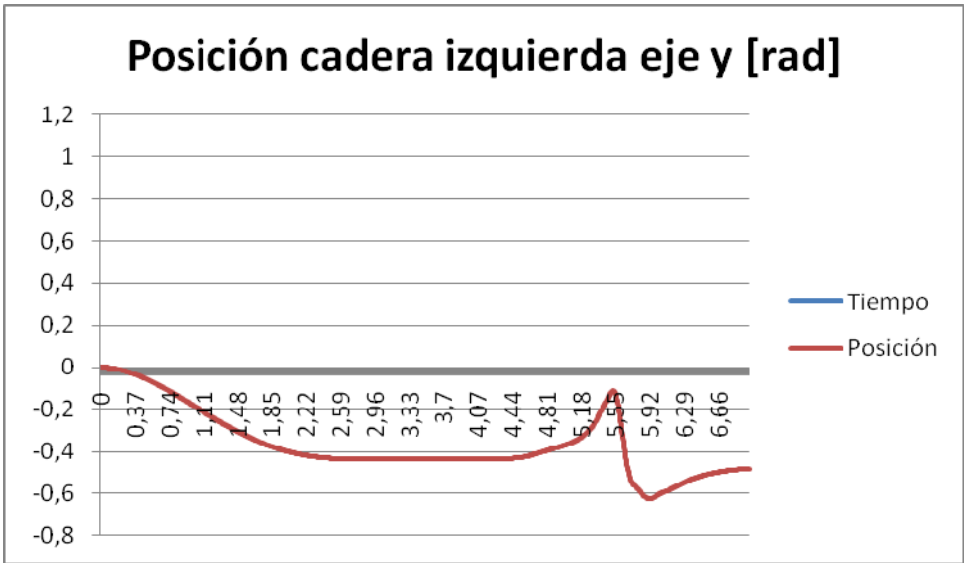


Figura 5.30 Gráfica posiciones de la cadera izquierda a lo largo de la simulación (eje y)

En 5.30 se obtiene la posición de la cadera izquierda en el eje y a lo largo de la simulación. Ésta comienza su movimiento desde el principio de la simulación, desplazándose durante los dos primeros segundos hacia delante, en el inicio del movimiento del paso. Se aprecia una variación brusca del valor de la posición entre los instantes 4.81 y 5.92 segundos, que son los correspondientes al tiempo que tarda el robot en realizar los dos pasos. Se puede comprobar que el inicio del valor de pico comienza durante el paso de la pierna derecha, y éste alcanza su valor mínimo (-0.6 rad) cuando el robot realiza el paso con la pierna izquierda y apoya ésta en el suelo. En la recta final inicia la recuperación de su posición inicial, aunque termina la simulación cuando su posición es de -0.5 rad.

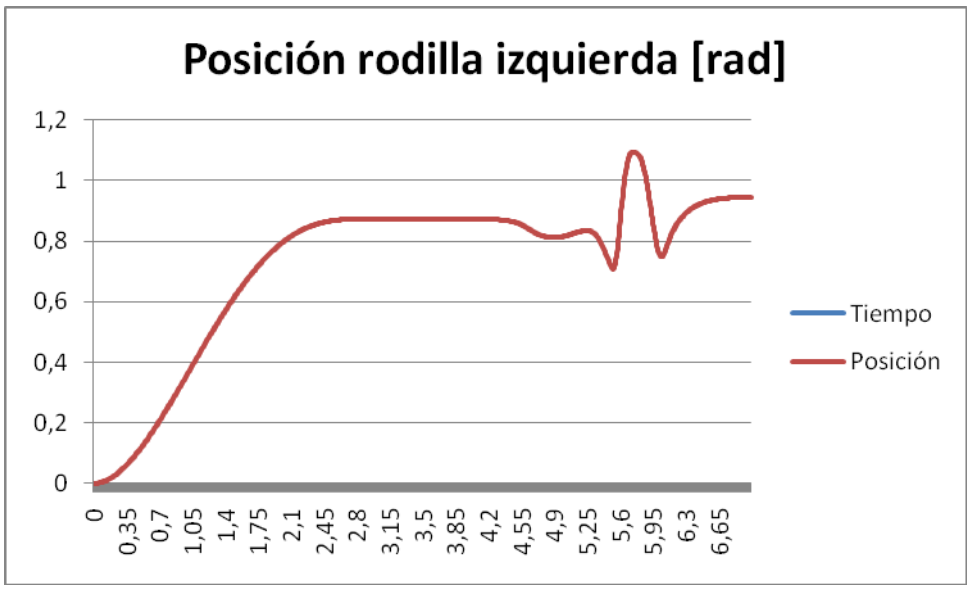


Figura 5.31 Gráfica posiciones de la rodilla izquierda a lo largo de la simulación

La Figura 5.31 muestra la posición de la rodilla izquierda a lo largo de la simulación. Ésta comienza su movimiento desde el principio de la simulación, desplazándose durante los primeros 2.5 segundos hacia atrás en lo que correspondería con el movimiento de flexión de las piernas, justo antes de empezar a dar los pasos. Permanece en esa posición hasta el segundo 4.8 en el que se aprecia una pequeña variación debido al paso de la pierna derecha. Entre los segundos 5.4 y 6 aproximadamente se produce una variación brusca del valor de la posición hasta llegar a alcanzar los 1.1 rad, lo que correspondería al giro de la rodilla durante el paso hacia delante de la pierna izquierda del robot. Posteriormente disminuye hasta la posición de 0.75 rad, para volver a aumentar hasta el final de la simulación llegando hasta los 0.95 rad.

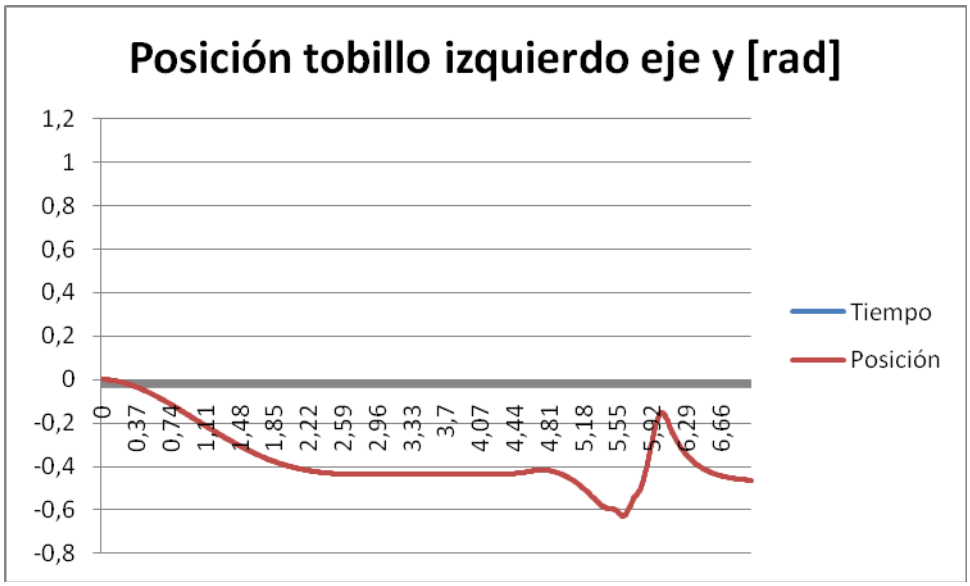


Figura 5.32 Gráfica posiciones del tobillo izquierdo a lo largo de la simulación (eje y)

En 5.32 se obtiene la posición del tobillo izquierdo en el eje y a lo largo de la simulación. Se desplaza desde el principio de la simulación durante los primeros 2.5 segundos hacia delante en lo que correspondería con el movimiento de flexión de las piernas, justo antes de empezar a dar los pasos. Permanece en esa posición hasta el segundo 4.8 en el que se aprecian dos picos de valores provocados por dos variaciones bruscas de posición hasta el instante de tiempo de 6 segundos. El primer pico se produce entre 4.8 y 5.5 segundos y es producido por el paso del robot con la otra pierna (la pierna derecha), llegando a alcanzar la posición de -0.6 rad. El segundo ocurre entre los instantes de tiempo de 5.5 segundos y 6 segundos, y es provocado por el paso con la pierna izquierda, en el que llega a alcanzar el valor aproximado de -0.18 rad. De aquí al final de la simulación disminuye su posición hasta el valor de -0.5 rad.

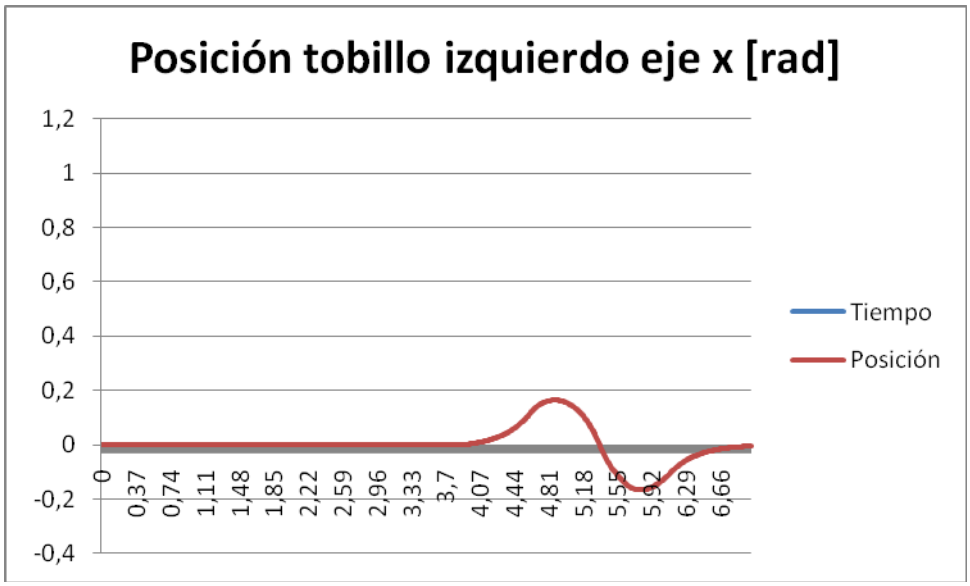


Figura 5.33 Gráfica posiciones del tobillo izquierdo a lo largo de la simulación (eje x)

En la Figura 5.33 se puede observar la posición del tobillo izquierdo en el eje x a lo largo de la simulación. El movimiento es totalmente análogo al del tobillo derecho en el eje x, por lo que no se explica de nuevo.

La velocidad, al igual que en el anterior caso, no juega un papel tan crucial como la posición en esta simulación en concreta, por lo que se vuelve a analizar la gráfica de velocidad sólo de la rodilla izquierda:

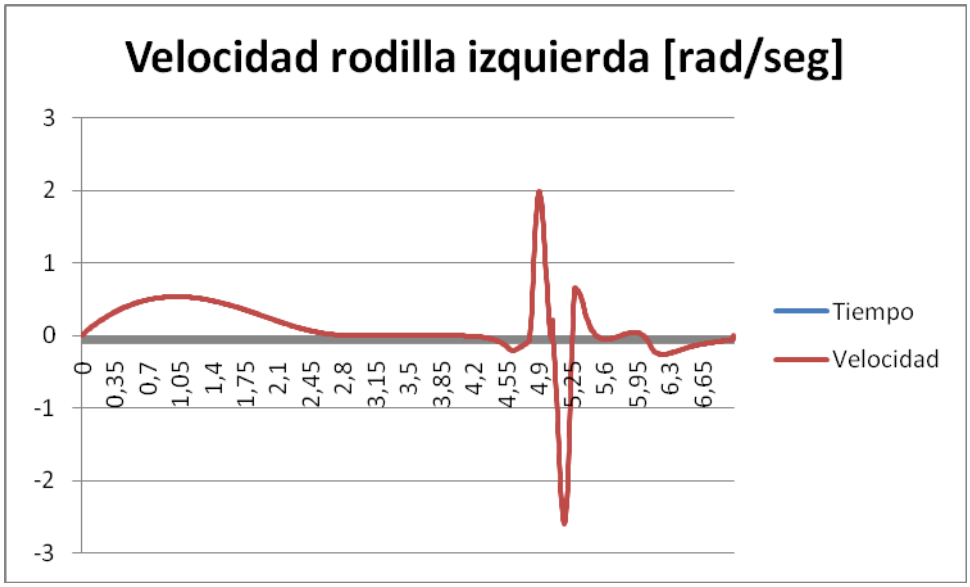


Figura 5.34 Gráfica velocidades de la rodilla izquierda a lo largo de la simulación

El resultado es idéntico al de la caminada inestable. La rodilla izquierda se mueve al principio realizando el movimiento de flexión de las piernas hasta alcanzar una velocidad de 0.5 rad/seg. El segundo tramo de movimientos se produce entre los segundos 4.55 y 6, donde tienen lugar los dos pasos que da el robot en la simulación. Alcanza unos valores de pico de 2 y -2.5 rad/seg, correspondientes al movimiento de la rodilla primero hacia atrás (sentido antihorario), y luego hacia delante (sentido horario).

5.4.3 Resultados de la simulación

A continuación se muestran diversas capturas obtenidas en la simulación. Se recomienda de nuevo comparar las diferentes figuras con las gráficas mostradas anteriormente, para asociar los diferentes movimientos del robot, con la variación de posiciones. Al igual que en el caso anterior en el CD adjunto a este proyecto se incluye un video de dicha simulación donde se puede comprobar la estabilidad de la caminada.

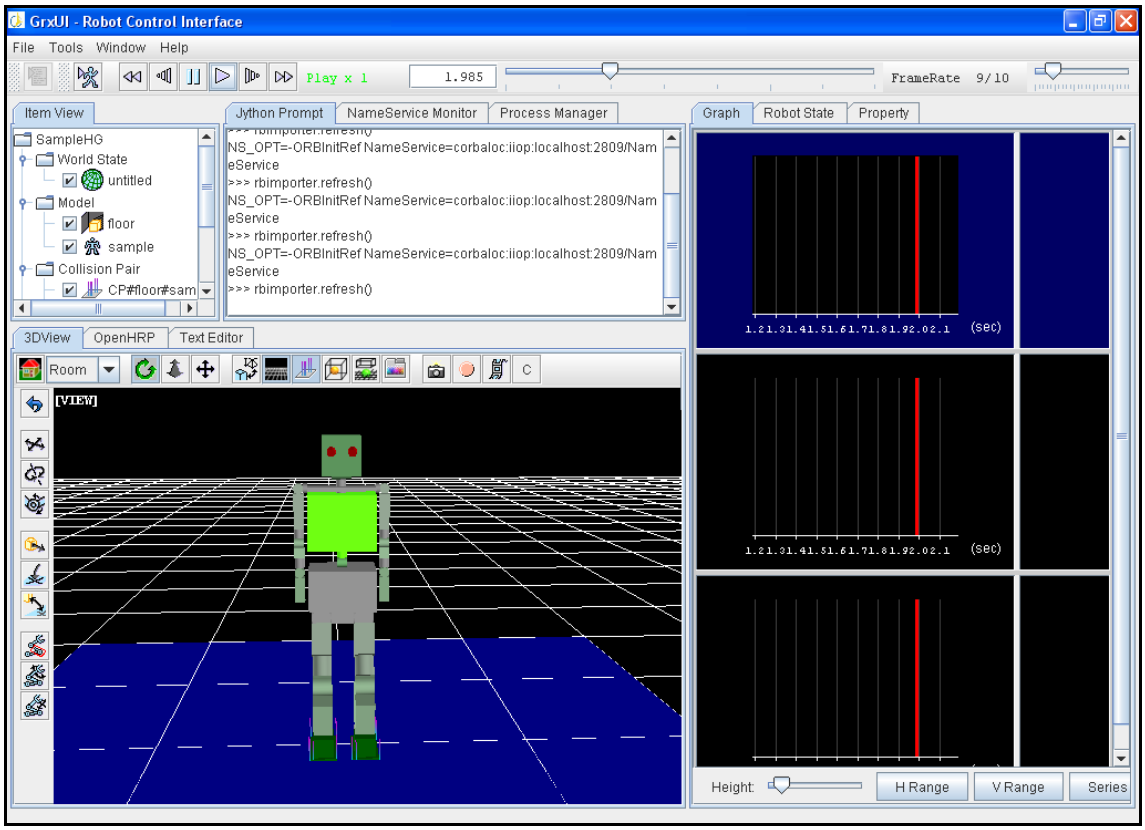


Figura 5.35 Inicio de la simulación. Trayectoria estable

Al igual que en el anterior caso, el robot se encuentra iniciando la flexión de las piernas para, una vez flexionadas rotar la cadera y comenzar el desplazamiento.

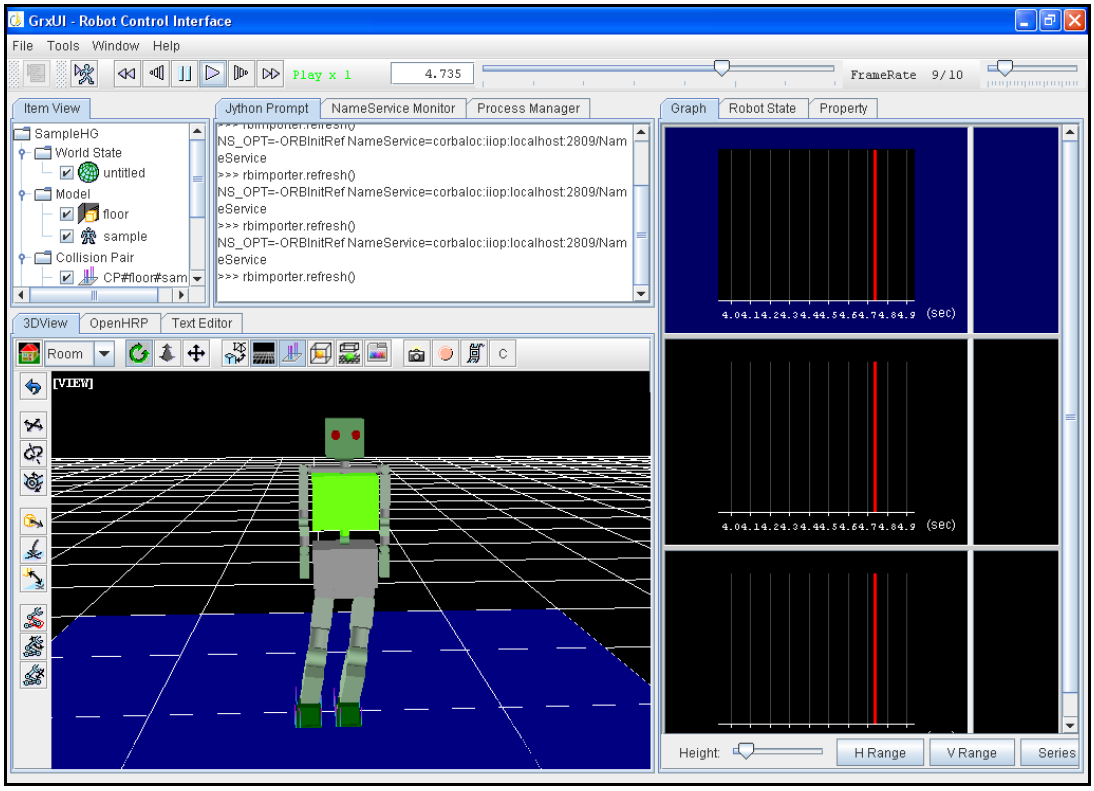


Figura 5.36 Inicio de la rotación de caderas. Trayectoria estable

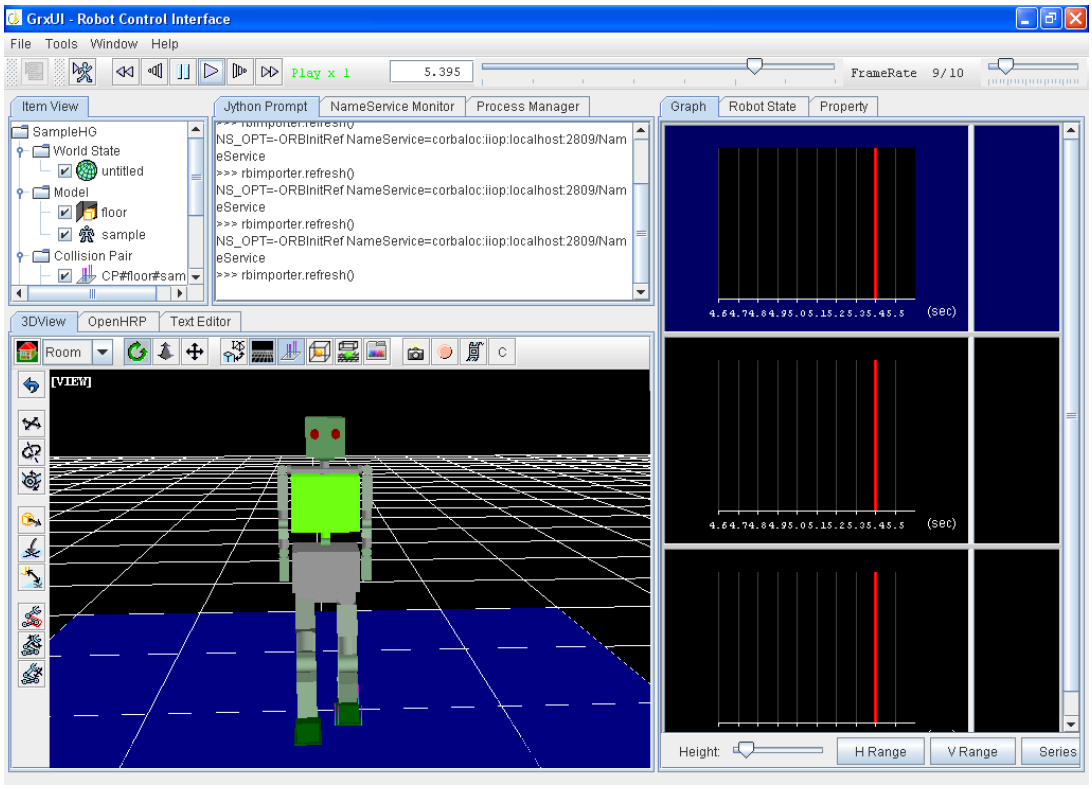


Figura 5.37 Primer paso del robot. Trayectoria estable

En la Figura 5.36 y la Figura 5.37 se puede observar como se ha corregido el desplazamiento lateral que había en el caso anterior, y realiza el primer paso sin desplazarse lateralmente de su posición central en el suelo. Además comienza el paso con la pierna correcta, tras desplazarse hacia la izquierda, mantiene la pierna izquierda como pierna de apoyo, y realiza el paso con la pierna derecha, lo que evita el desequilibrio que se producía en el primer paso en el anterior caso.

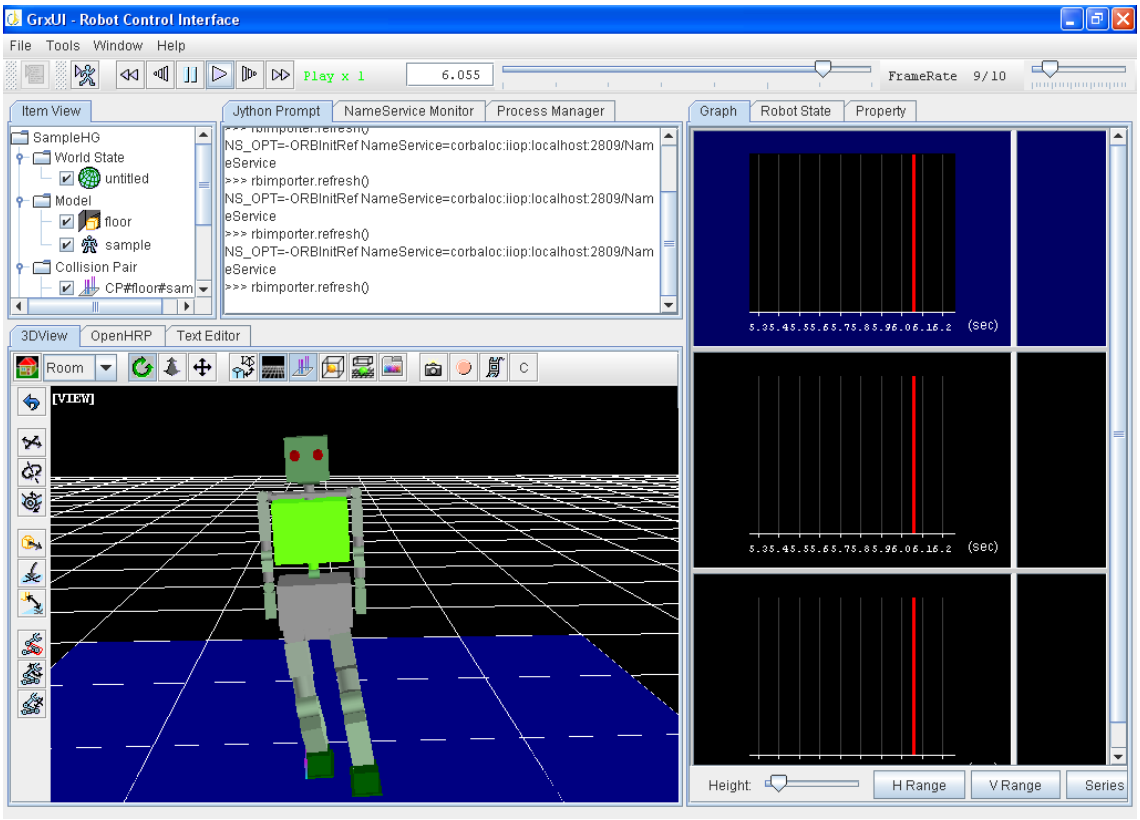


Figura 5.38 Segundo paso del robot. Trayectoria estable

El segundo paso lo realiza también sin ningún desplazamiento inadecuado que pueda provocar la desestabilización del sistema.

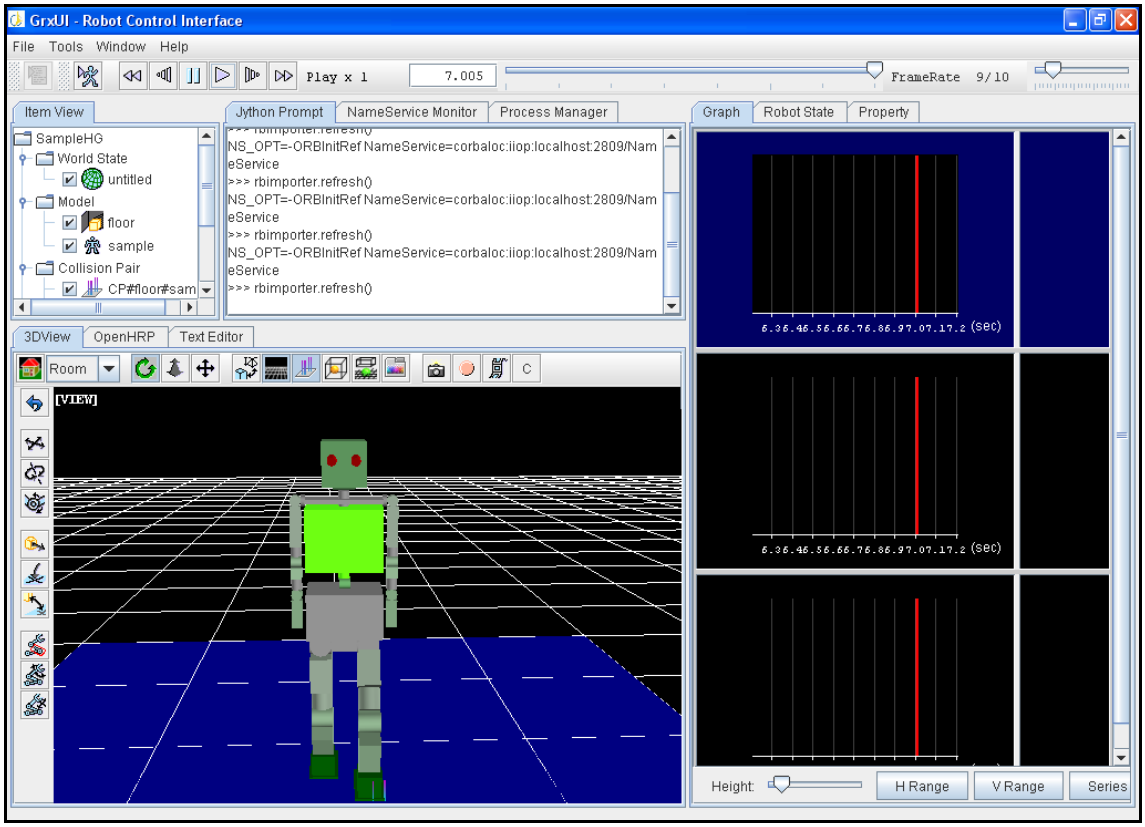


Figura 5.39 Final de la simulación. Trayectoria estable

El robot ha terminado la simulación erguido, sin ningún signo de desequilibrio, lo que valida la trayectoria obtenida, además de la fiabilidad del simulador en cuanto al modelo introducido del robot Rh-2, ya que el resultado obtenido es igual al que se esperaba obtener.

5.5 Comparación de resultados

En este apartado se va a realizar un análisis más a fondo de las principales diferencias de ambas trayectorias, para determinar las causas que provocan que una trayectoria sea inestable y la otra estable.

5.5.1 Comparación en la pierna derecha

A continuación se muestran los datos que estaban correctos en las dos simulaciones y por tanto, no han sido objeto de cambio, de tal forma que se pueda comprobar que sus datos de trayectorias no han sido modificados:

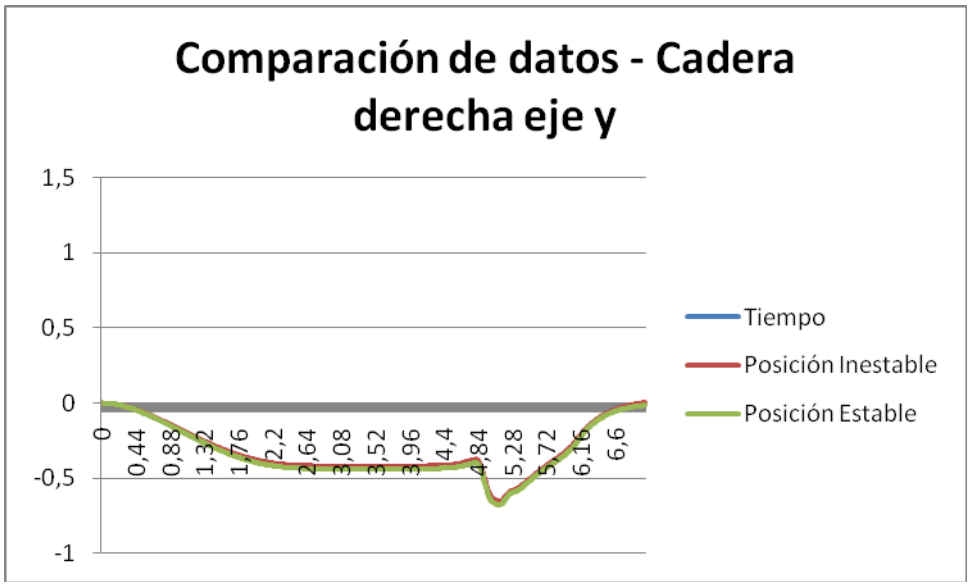


Figura 5.40 Comparación cadera derecha eje y

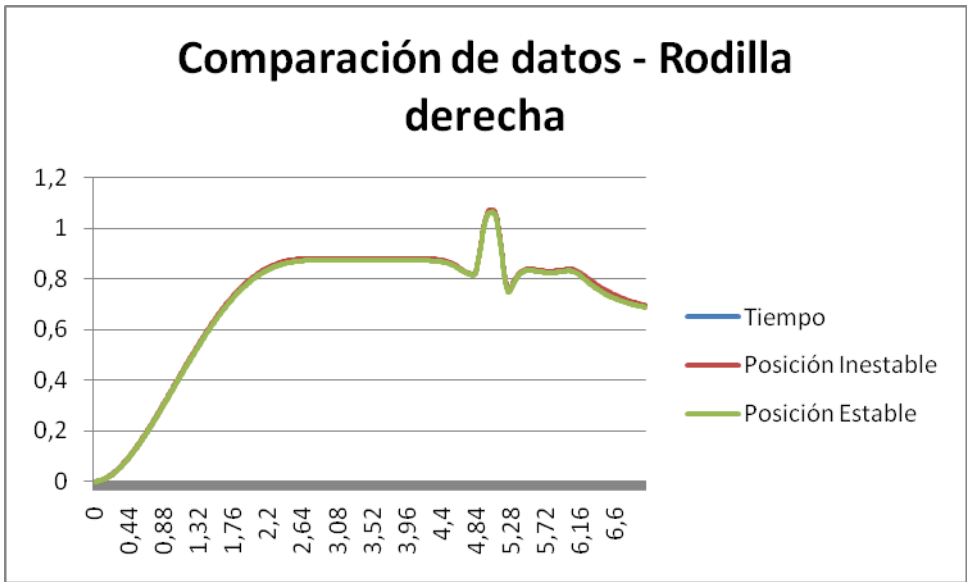


Figura 5.41 Comparación rodilla derecha

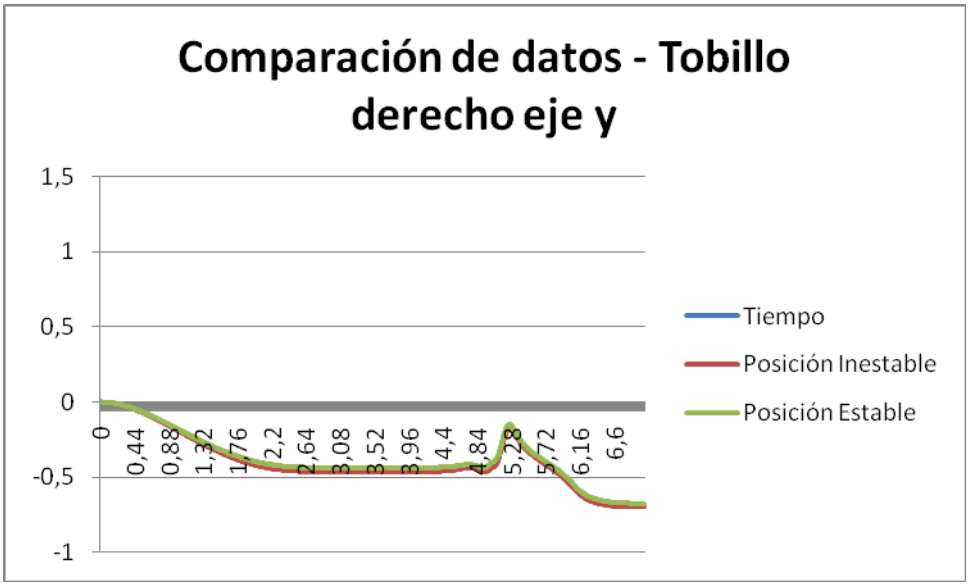


Figura 5.42 Comparación tobillo derecho eje y

Las articulaciones que han cambiado su valor en cuanto a posición en las trayectorias introducidas son la cadera derecha en el eje x (Figura 5.43), y el tobillo derecho en el eje x (Figura 5.44). Cabe destacar que estas variaciones tienen todo el sentido, debido a que la inestabilidad era debida a desplazamientos en el eje x inapropiados, además de lo comentado de la pierna de apoyo. La variación de posiciones en la cadera y en el tobillo han proporcionado la estabilidad en la caminada al modelo.

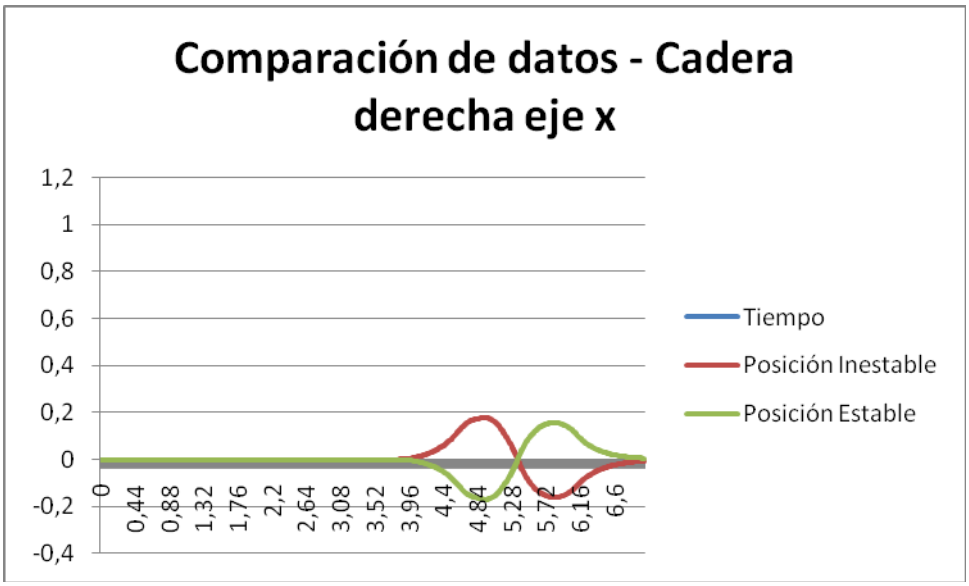


Figura 5.43 Comparación cadera derecha eje x

Se observa un cambio de movimiento en cuanto a sentido de giro en el momento de realizar los pasos de la caminada del robot. La cadera en el eje x con la trayectoria estable se desplaza hacia la derecha, lo que permite respecto de la otra trayectoria realizar el primer paso de la simulación con el pie de apoyo bien situado, dando estabilidad al robot. La variación ha sido invertir el sentido de giro de la cadera en el eje x en el momento de los pasos del robot.

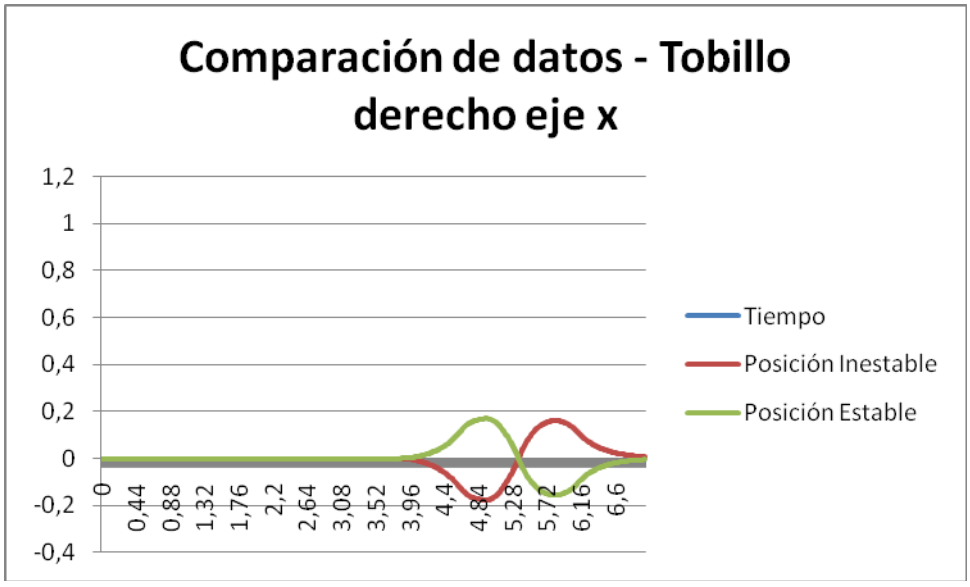


Figura 5.44 Comparación tobillo derecho eje x

En el tobillo en el eje x (Figura 5.44) pasa algo muy parecido a lo comentado anteriormente. Se comprueba que el movimiento del tobillo tiene que ser inverso al de la cadera para mantener el pie completamente apoyado en el suelo en el momento de giro de la cadera. Por ello al cambiar la trayectoria y el sentido de giro de la cadera, el tobillo también se modifica, de tal forma que quede de forma inversa a la cadera.

5.5.2 Comparación en la pierna izquierda

A continuación, al igual que en el apartado anterior, primero se muestran los datos que estaban correctos en las dos simulaciones y por tanto, no han sido objeto de cambio, de tal forma que se pueda comprobar que sus datos de trayectorias no han sido modificados:

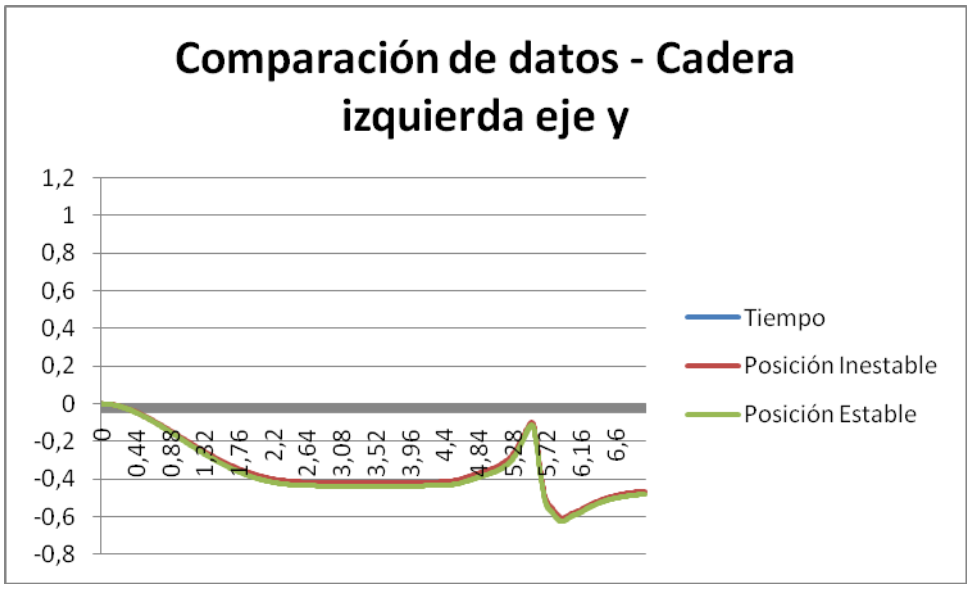


Figura 5.45 Comparación cadera izquierda eje y

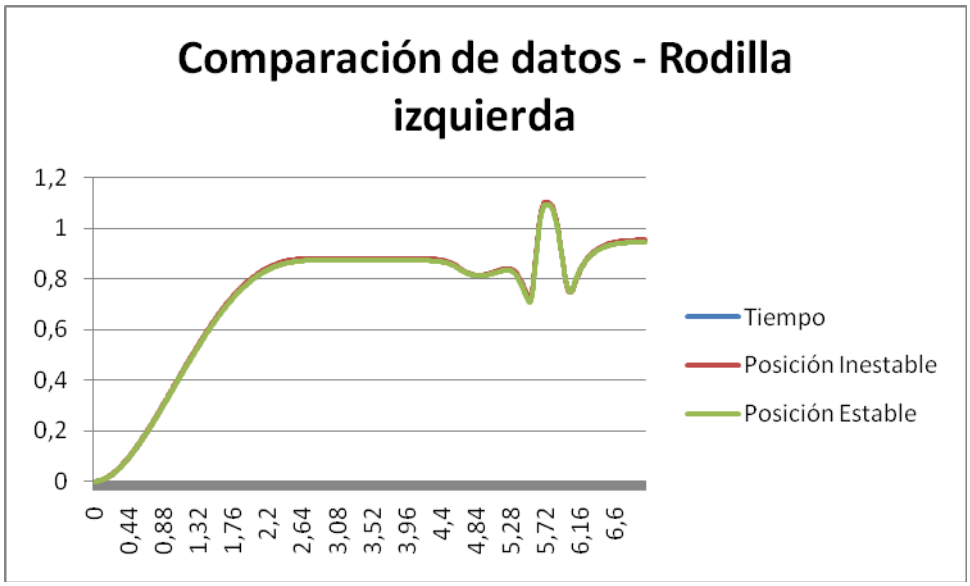


Figura 5.46 Comparación rodilla izquierda

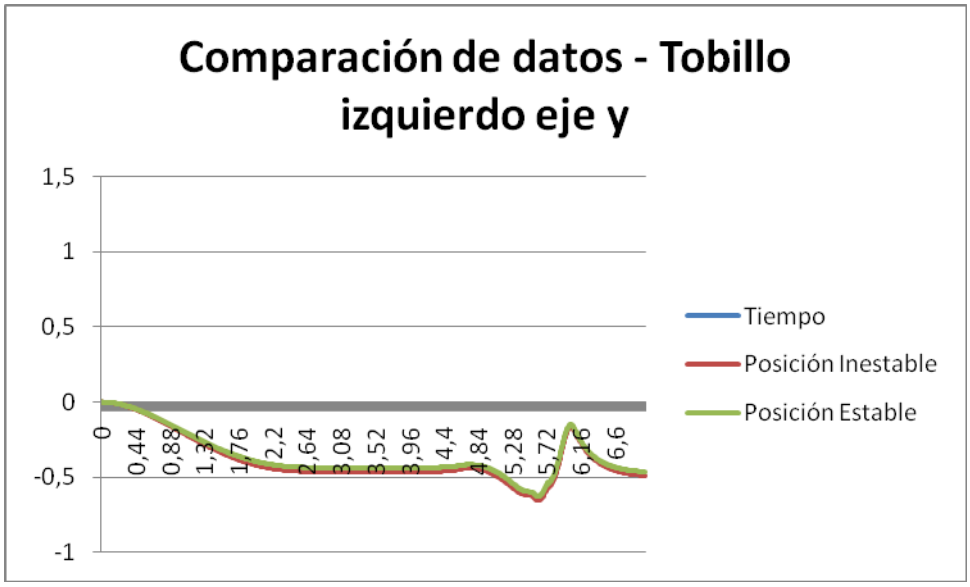


Figura 5.47 Comparación tobillo izquierdo eje y

Como en el apartado anterior, las articulaciones que han cambiado su valor en cuanto a posición en las trayectorias introducidas son la cadera izquierda en el eje x (Figura 5.48), y el tobillo derecho en el eje x (Figura 5.49).

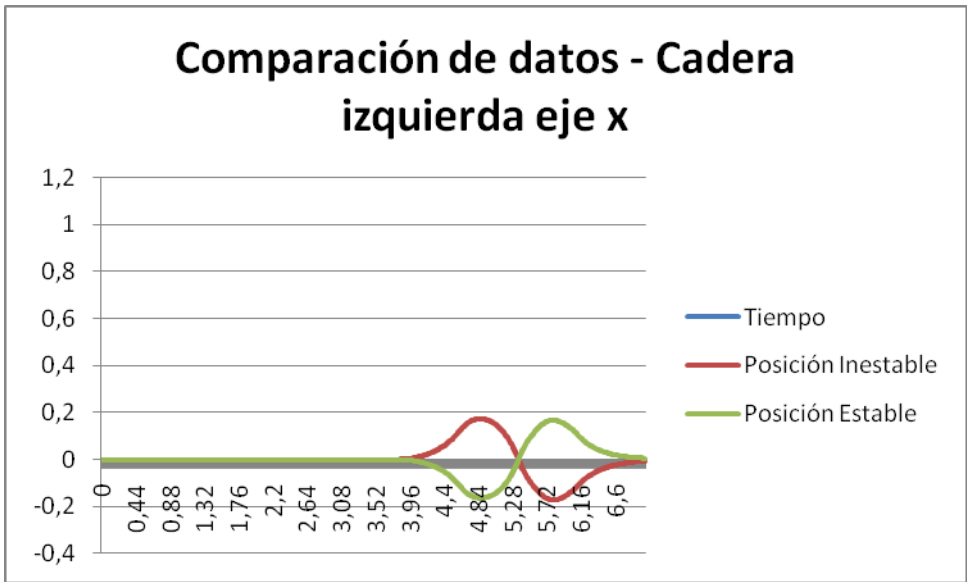


Figura 5.48 Comparación cadera izquierda eje x

Como en la pierna derecha, se observa un cambio de movimiento en cuanto a sentido de giro en el momento de realizar los pasos de la caminada del robot. La cadera en el eje x con la trayectoria estable se desplaza hacia la derecha, lo que permite respecto de la otra trayectoria realizar el primer paso de la simulación con el pie de apoyo bien situado, dando estabilidad al robot. La variación ha sido invertir el sentido de giro de la cadera en el eje x en el momento de los pasos del robot.

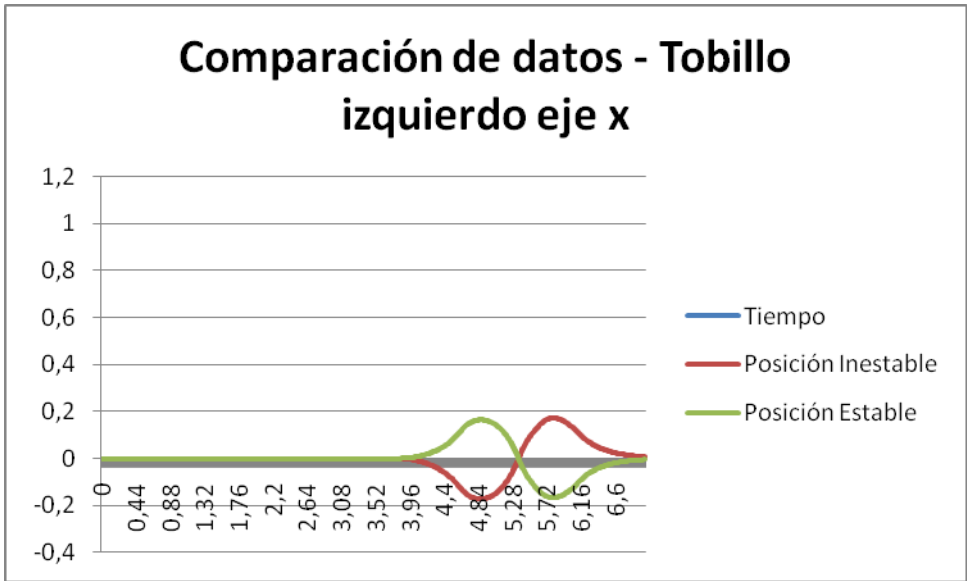


Figura 5.49 Comparación tobillo izquierdo eje x

En el tobillo en el eje x (Figura 5.49) como se expuso para el movimiento de la pierna derecha el movimiento es inverso al de la cadera para mantener el pie completamente apoyado en el suelo en el momento de giro de la cadera. Por ello al cambiar la trayectoria y el sentido de giro de la cadera, el tobillo también se modifica, de tal forma que quede de forma inversa a la cadera.

6 Conclusiones

El principal objetivo de este proyecto consistía en el modelado de la plataforma robótica Rh-2 en el simulador OpenHRP3. Dicho modelado permitiría la simulación de tareas del robot Rh-2 en el entorno virtual, además de proporcionar datos físicos necesarios para el desarrollo del prototipo del robot, como por ejemplo los pares máximos de fuerza que soporta en una determinada tarea.

Se partió del estudio en profundidad del simulador OpenHRP3, y de todos los componentes que requería para poder realizar una simulación. Una vez comprendido su funcionamiento, se ha desarrollado el modelo VRML del robot Rh-2 para la plataforma OpenHRP3. Este modelo ha sido realizado siguiendo todas las especificaciones físicas que se espera que tenga el robot (geometría, peso, etc.).

Después de desarrollar el modelo VRML se han programado distintas tareas en el simulador. Estas tareas corresponden a una caminada estable e inestable del robot, a través de unas trayectorias proporcionadas, en las que se valida el comportamiento del simulador ante factores que provoquen la desestabilización del sistema de control.

La conclusión final del proyecto es la validación de las tareas programadas en el simulador, así como del modelo VRML desarrollado para la plataforma de simulación OpenHRP3. Se ha comprobado que las trayectorias desarrolladas mediante otras plataformas son también válidas para el modelo desarrollado, lo que implica la validación del mismo, así como la del propio simulador, debido a que el modelo responde según los resultados esperados.

7 Trabajos futuros

Como desarrollos futuros pueden incluirse las siguientes líneas de estudio:

- Probar el modelo con más trayectorias de casos ya desarrollados, sería importante comprobar que sigue comportándose según lo esperado ante casos más sofisticados.

- El simulador consta de una caja negra, que es el controlador total del sistema, éste actúa estabilizando el modelo, aunque no se le hayan incluido márgenes de error u offset. Descubrir dónde se ubica y su funcionamiento sería una gran línea de investigación futura.

- Comprobar el funcionamiento del simulador con otros robots humanoides ya diseñados, para verificar que los resultados se asemejan, de la misma forma que se realizó con el robot HOAP-3.

- Mejorar el fichero VRML tanto del modelo del robot como de su entorno. Se puede perfeccionar la apariencia física del robot para que sea lo más semejante al robot Rh-2. El diseño realizado en este proyecto es muy parejo al que venía por defecto con el paquete del simulador, por lo tanto un cambio de apariencia le daría todavía más realismo a la simulación.

8 Bibliografía

MANUALES

[1] “OpenHRP3 Course”. General Robotix Inc, 2008.

LIBROS

[2] “3D modeling and animation: synthesis and analysis techniques for the human body”, Nikos Sarris,Michael G. Strintzis.

TESIS DE MÁSTER

[3] “Plataforma de simulación cinemática y dinámica de futuras versiones del robot Asibot”, Tesis de Máster de Carlos Pérez de la Fuente. Noviembre 2008.

PROYECTOS FIN DE CARRERA

[4] “Simulación de la plataforma robótica HOAP-3 en el simulador OpenHRP3”, Proyecto Fin de Carrera de Tamara Ramos Cambero. Septiembre 2009.

[5] “Caminata del robot humanoide Rh-2 en la plataforma de simulación OpenHRP”, Proyecto Fin de Carrera de Carlos de Torre Doblas. Julio 2010.

DIRECCIONES DE INTERNET

[6] http://www.web3d.org/

[7] http://www.uc3m.es/

[8] http://www.openrtp.jp

[9] http://www.is.aist.go.jp

[10] http://www.microsoft.com

[11] http://www.boost.org

[12] http://www.netlib.org/clapack

[13] http://tvmet.sourceforge.net

[14] http://www.wikipedia.com

[15] http://omniorb.sourceforge.net

[16] http://www.python.org

[17] http://www.java.com

[18] http://www.jython.org

9 Anexos

9.1 Código VRML del modelo del robot Rh-2

```
#VRML V2.0 utf8
#-------------------------------------------------------------
# OpenHRP Sample Model
#
# author    Ichitaro Kohara (YNL, Univ. of Tokyo)
# version   1.0 (2000.11.08)
# modified  Hirohisa Hirukawa (ETL)
# version   1.1 (2000.11.24)
# modified  Natsuki Miyata (MEL)
# version   1.1 (2000.12.7)
#-------------------------------------------------------------

PROTO Joint [
  exposedField     SFVec3f      center              0 0 0
  exposedField     MFNode       children            []
  exposedField     MFFloat      llimit              []
  exposedField     MFFloat      lvlimit             []
  exposedField     SFRotation   limitOrientation    0 0 1 0
  exposedField     SFString     name                ""
  exposedField     SFRotation   rotation            0 0 1 0
  exposedField     SFVec3f      scale               1 1 1
  exposedField     SFRotation   scaleOrientation    0 0 1 0
  exposedField     MFFloat      stiffness           [ 0 0 0 ]
  exposedField     SFVec3f      translation         0 0 0
  exposedField     MFFloat      ulimit              []
  exposedField     MFFloat      uvlimit             []
  exposedField     SFString     jointType           ""
  exposedField     SFInt32      jointId             -1
  exposedField     SFString     jointAxis           "Z"

  exposedField     SFFloat      gearRatio           1
  exposedField     SFFloat      rotorInertia        0
  exposedField     SFFloat      rotorResistor       0
  exposedField     SFFloat      torqueConst         1
  exposedField     SFFloat      encoderPulse        1
]
{
  Transform {
    center           IS center
    children         IS children
    rotation         IS rotation
    scale            IS scale
    scaleOrientation IS scaleOrientation
    translation      IS translation
  }
}

PROTO Segment [
  field           SFVec3f     bboxCenter        0 0 0
  field           SFVec3f     bboxSize          -1 -1 -1
  exposedField    SFVec3f     centerOfMass      0 0 0
  exposedField    MFNode      children          [ ]
  exposedField    SFNode      coord             NULL
  exposedField    MFNode      displacers        [ ]
  exposedField    SFFloat     mass              0
```

```
  exposedField    MFFloat     momentsOfInertia  [ 0 0 0 0 0 0 0 0 0 ]
  exposedField    SFString    name              ""
  eventIn         MFNode      addChildren
  eventIn         MFNode      removeChildren
]
{
  Group {
    addChildren    IS addChildren
    bboxCenter     IS bboxCenter
    bboxSize       IS bboxSize
    children       IS children
    removeChildren IS removeChildren
  }
}

PROTO Humanoid [
  field           SFVec3f    bboxCenter            0 0 0
  field           SFVec3f    bboxSize              -1 -1 -1
  exposedField    SFVec3f    center                0 0 0
  exposedField    MFNode     humanoidBody          [ ]
  exposedField    MFString   info                  [ ]
  exposedField    MFNode     joints                [ ]
  exposedField    SFString   name                  ""
  exposedField    SFRotation rotation              0 0 1 0
  exposedField    SFVec3f    scale                 1 1 1
  exposedField    SFRotation scaleOrientation      0 0 1 0
  exposedField    MFNode     segments              [ ]
  exposedField    MFNode     sites                 [ ]
  exposedField    SFVec3f    translation           0 0 0
  exposedField    SFString   version               "1.1"
  exposedField    MFNode     viewpoints            [ ]
]
{
  Transform {
    bboxCenter       IS bboxCenter
    bboxSize         IS bboxSize
    center           IS center
    rotation         IS rotation
    scale            IS scale
    scaleOrientation IS scaleOrientation
    translation      IS translation
    children [
      Group {
        children IS viewpoints
      }
      Group {
        children IS humanoidBody
      }
    ]
  }
}

PROTO VisionSensor [
  exposedField SFVec3f    translation       0 0 0
  exposedField SFRotation rotation          0 0 1 0
  exposedField MFNode     children          [ ]
  exposedField SFFloat    fieldOfView       0.785398
  exposedField SFString   name              ""
  exposedField SFFloat    frontClipDistance 0.01
  exposedField SFFloat    backClipDistance  10.0
  exposedField SFString   type              "NONE"
```

```
  exposedField SFInt32    sensorId          -1
  exposedField SFInt32    width             320
  exposedField SFInt32    height            240
  exposedField SFFloat    frameRate         30
]
{
  Transform {
    rotation         IS rotation
    translation      IS translation
    children         IS children
  }
}


PROTO ForceSensor [
  exposedField SFVec3f    maxForce    -1 -1 -1
  exposedField SFVec3f    maxTorque   -1 -1 -1
  exposedField SFVec3f    translation 0 0 0
  exposedField SFRotation rotation    0 0 1 0
  exposedField SFInt32    sensorId    -1
]
{
  Transform {
    translation IS translation
    rotation    IS rotation
  }
}

PROTO Gyro [
  exposedField SFVec3f    maxAngularVelocity -1 -1 -1
  exposedField SFVec3f    translation        0 0 0
  exposedField SFRotation rotation           0 0 1 0
  exposedField SFInt32    sensorId           -1
]
{
  Transform {
    translation IS translation
    rotation    IS rotation
  }
}

PROTO AccelerationSensor [
  exposedField SFVec3f    maxAcceleration -1 -1 -1
  exposedField SFVec3f    translation     0 0 0
  exposedField SFRotation rotation        0 0 1 0
  exposedField SFInt32    sensorId        -1
]
{
  Transform {
    translation IS translation
    rotation    IS rotation
  }
}

PROTO PressureSensor [
  exposedField SFFloat    maxPressure -1
  exposedField SFVec3f    translation 0 0 0
  exposedField SFRotation rotation    0 0 1 0
  exposedField SFInt32    sensorId    -1
]
{
```

```
  Transform {
    translation IS translation
    rotation    IS rotation
  }
}

PROTO PhotoInterrupter [
  exposedField SFVec3f transmitter 0 0 0
  exposedField SFVec3f receiver    0 0 0
  exposedField SFInt32 sensorId    -1
]
{
  Transform{
    children [
      Transform{
        translation IS transmitter
      }
      Transform{
        translation IS receiver
      }
    ]
  }
}

PROTO CylinderSensorZ [
    exposedField    SFFloat    maxAngle       -1
    exposedField    SFFloat    minAngle        0
    exposedField    MFNode     children        [ ]
]
{
  Transform{
    rotation 1 0 0 1.5708
    children [
      DEF SensorY CylinderSensor{
        maxAngle IS maxAngle
        minAngle IS minAngle
      }
      DEF AxisY Transform{
        children [
          Transform{
            rotation 1 0 0 -1.5708
            children IS children
          }
        ]
      }
    ]
  }
  ROUTE SensorY.rotation_changed TO AxisY.set_rotation
}

PROTO CylinderSensorY [
    exposedField    SFFloat    maxAngle       -1
    exposedField    SFFloat    minAngle        0
    exposedField    MFNode     children        [ ]
]
{
  Transform{
    rotation 0 1 0 1.5708
    children [
      DEF SensorX CylinderSensor{
        maxAngle IS maxAngle
```

```
        minAngle IS minAngle
      }
      DEF AxisX Transform{
        children [
          Transform{
            rotation 0 1 0 -1.5708
            children IS children
          }
        ]
      }
    ]
  }
  ROUTE SensorX.rotation_changed TO AxisX.set_rotation
}

PROTO CylinderSensorX [
    exposedField    SFFloat    maxAngle       -1
    exposedField    SFFloat    minAngle        0
    exposedField    MFNode     children        [ ]
]
{
  Transform{
    rotation 0 0 1 -1.5708
    children [
      DEF SensorZ CylinderSensor{
        maxAngle IS maxAngle
        minAngle IS minAngle
      }
      DEF AxisZ Transform{
        children [
          Transform{
            rotation 0 0 1 1.5708
            children IS children
          }
        ]
      }
    ]
  }
  ROUTE SensorZ.rotation_changed TO AxisZ.set_rotation
}

NavigationInfo {
  avatarSize    0.5
  headlight     TRUE
  type  ["EXAMINE", "ANY"]
}

Background {
  skyColor 0.4 0.6 0.4
}

Viewpoint {
  position    3 0 0.835
  orientation 0.5770 0.5775 0.5775 2.0935
}

DEF SampleRobot Humanoid {
  name "sample"
  version "1.1"
  info [
    "This is a sample model of OpenHRP."
```

```
  "You can modify and use this model freely."
  "Author  : Ichitaro Kohara, YNL, Univ. of Tokyo"
  "Date    : 2000.11.08"
  "Modifying Author  : Natsuki Miyata, MEL"
  "Date    : 2000.12.08"
  "Version : 1.1"
]

humanoidBody [

  DEF WAIST Joint {
    jointType "free"
    translation 0 0 0.76975
    children [
    DEF sensor0 ForceSensor { sensorId 0 }
      DEF gsensor AccelerationSensor { sensorId 0 }
      DEF gyrometer Gyro { sensorId 0 }
      DEF WAIST_LINK0 Segment {
        centerOfMass  0 0 0.0375
        mass    27.0
        momentsOfInertia [ 1 0 0 0 1 0 0 0 1 ]
        children [
          Shape {
            appearance DEF WAIST_APP Appearance {
              material Material {
              }
            }
            geometry Box { size 0.08 0.11754 0.08 }
          }
          Transform {
            translation   0 0 0.144
            children      Shape {
              appearance  USE WAIST_APP
              geometry Box { size 0.268 0.34 0.208 }
            }
          }
        ]
      }

      DEF WAIST_P Joint {


      jointType "rotate"
        jointAxis "Y"

        jointId  24
         translation     0 0 0.273
        children [
        DEF sensor1 ForceSensor { sensorId 1 }
          DEF WAIST_LINK1 Segment {
            centerOfMass  0 0 -0.1
            mass    6.0
            momentsOfInertia [ 1 0 0 0 1 0 0 0 1 ]
            children [
              Shape {
                appearance  DEF WAIST_LINK1_APP Appearance {
                  material  Material {
                    diffuseColor   0.6 1.0 0.6
                  }
                }
                geometry Cylinder { radius 0.025 height 0.05 }
```

```
        }
      ]
    }
    DEF WAIST_Y Joint {
      jointType "rotate"
    jointAxis "Z"
      jointId  25
      children [
      DEF sensor2 ForceSensor { sensorId 2 }
        DEF WAIST_LINK2 Segment {
          centerOfMass  0.11 0 0.25
          mass    30.0
          momentsOfInertia [ 1 0 0 0 1 0 0 0 1 ]
          children [
            Transform {
           rotation 1 0 0 1.570
           translation 0 0 0.05
              children Shape {
                appearance DEF WAIST_LINK2_APP Appearance {
                  material      Material {
                    diffuseColor       0.6 1.8 0.1
                  }
                }
                geometry Cylinder { radius 0.025 height 0.05 }
              }
            }
            Transform {
              translation 0 0 0.2125
              children  Shape {
                appearance USE WAIST_LINK2_APP
                geometry Box { size 0.268 0.34 0.275 }
              }
            } # Transform
          ]
        } # segment WAIST_LINK2

        DEF CHEST_Y Joint {
          jointType "rotate"
          jointId  26
          translation 0 0 0.383015
          children [
          DEF sensor3 ForceSensor { sensorId 3 }
            DEF WAIST_LINK3 Segment {
              centerOfMass  0 0 0
              mass    13.0
              momentsOfInertia [ 1 0 0 0 1 0 0 0 1 ]
              children [
                Transform {
                  rotation 1 0 0 1.5708
                  children Shape {
                    appearance DEF WAIST_LINK3_APP Appearance
{
                      material  Material {
                        diffuseColor 0.8 0.8 0.8
                      }
                    }
                    geometry Cylinder { radius 0.025 height
0.06603 }
                  }
                }
                Transform {
```

```
                    translation 0 0 -0.01
                    children  Shape {
                     appearance USE WAIST_LINK3_APP
                        geometry Box { size 0.15 0.34 0.02 }
                    }

                  } # Transform



                    ]
                    }

                  DEF CHEST_P Joint {
                jointType "rotate"
                jointAxis "Y"
                jointId  27
                translation 0 0 -0.006
                children [
                DEF sensor4 ForceSensor { sensorId 4 }
                  DEF WAIST_LINK4 Segment {
                    centerOfMass  0 0 0
                    mass    13.0
                    momentsOfInertia [ 1 0 0 0 1 0 0 0 1 ]
                    children [
                      Transform {

                        translation 0 0 0.065
                        children  Shape {
                          appearance      Appearance {
                            material      Material {
                              diffuseColor  0.5 0.8 0.5
                            }
                          }
                          geometry Cylinder { radius 0.025 height
0.1 }
                        }
                      }
                      Transform {
                        translation -0.015 0 0.16
                        children  Shape {
                          appearance      Appearance {
                            material      Material {
                              diffuseColor 0.5 0.8 0.5
                            }
                          }
                          geometry Box { size 0.31 0.19 0.19 }
                        } # shape
                      } # transform
                    ]
                  } # segment WAIST_LINK4

                  DEF VISION_SENSOR1 VisionSensor {
                    translation   0.18 0.05 0.18
                    rotation      0.4472 -0.4472 -0.7746 1.8235
                    name          "LeftCamera"
                    type          "DEPTH"
                    sensorId      0
                    children [
                    DEF sensor5 ForceSensor { sensorId 5 }
                      DEF CAMERA_SHAPE Transform {
```

```
        rotation  1 0 0 -1.5708
        children [
          Shape {
            geometry Cylinder {
              radius      0.02
              height      0.025
            }
            appearance Appearance {
              material Material {
                diffuseColor      1 0 0
              }
            }
          }
        ]
      }
    ]
  }
  DEF VISION_SENSOR2 VisionSensor {
    translation   0.18 -0.05 0.18
    rotation      0.4472 -0.4472 -0.7746 1.8235
    name          "RightCamera"
    type          "DEPTH"
    sensorId      1
    children [
      USE CAMERA_SHAPE
    ]
  }

 #==================== Left Arm ====================

  DEF LARM_SHOULDER_P Joint {
    jointType "rotate"
  jointAxis "Y"
    jointId  18
    translation   0 0.215 -0.005
    children [
    DEF sensor6 ForceSensor { sensorId 6 }
      DEF LARM_LINK1 Segment {
        centerOfMass  0.1 0 0
        mass    3.0
        momentsOfInertia [ 1 0 0 0 1 0 0 0 1 ]
        children Transform {
          translation 0 -0.035 0
          children DEF ARM_SHAPE1 Shape {
            appearance Appearance {
              material Material {
              }
            }
            geometry Cylinder { radius 0.025 height
0.02 }
          }
        }
      }

      DEF LARM_SHOULDER_R Joint {
        jointType "rotate"
  jointAxis "X"
        jointId  19
        children [
        DEF sensor7 ForceSensor { sensorId 7 }
          DEF LARM_LINK2 Segment {
```

```
                        centerOfMass  0 0 -0.1
                        mass    0.6
                        momentsOfInertia [ 1 0 0 0 1 0 0 0 1 ]
                        children DEF ARM_SHAPE2 Transform {
                          children [
                            Transform {
                              rotation 0 0 1 1.5708
                              children Shape {
                                appearance DEF ARM_LINK2_APP
Appearance {
                                  material  Material {
                                    diffuseColor 0.8 0.9 0.8
                                  }
                                }
                                geometry Cylinder { radius 0.025
height 0.05 }
                              }
                            }
                            Transform {
                              translation   0 0 -0.1140875
                              children      Shape {
                                appearance USE ARM_LINK2_APP
                                geometry Box { size 0.05 0.05
0.178175 }
                              }
                            }
                          ]
                        }
                      } # Segment LARM_LINK2

                      DEF LARM_SHOULDER_Y Joint {
                        jointType "rotate"
                        jointId  20
                        translation     0 0 -0.271825
                        children [
                        DEF sensor8 ForceSensor { sensorId 8 }
                          DEF LARM_LINK3 Segment {
                            centerOfMass  0 0 0
                            mass    1.0
                            momentsOfInertia [ 1 0 0 0 1 0 0 0 1
]
                            children DEF ARM_SHAPE3 Transform {
                              translation 0 0 0.05
                              rotation 1 0 0 1.5708
                              children Shape {
                                appearance Appearance {
                                  material Material {
                                  }
                                }
                                geometry Cylinder { radius 0.025
height 0.05}
                              }
                            }
                          } # Segment LARM_LINK3

                          DEF LARM_ELBOW Joint {
                            jointType "rotate"
                    jointAxis "Y"
                            jointId  21
                            children [
```

```
                                DEF sensor9 ForceSensor { sensorId 9
}
                                  DEF LARM_LINK4 Segment {
                                    centerOfMass  0 0 -0.3
                                    mass    0.6
                                    momentsOfInertia [ 1 0 0 0 1 0 0
0 1 ]
                                    children DEF ARM_SHAPE4
Transform {
                                      children [
                                        Shape {
                                          appearance DEF ARM_APP4
Appearance {
                                            material    Material {
                                              diffuseColor     0.8
1.0 0.8
                                            }
                                          }
                                          geometry Cylinder { radius
0.025 height 0.05 }
                                        }
                                        Transform {
                                          translation  0 0 -
0.05079375
                                          children        Shape {
                                            appearance USE ARM_APP4
                                            geometry Box { size 0.05
0.05 0.0515875 }
                                          }
                                        }
                                      ]
                                    }
                                  } # Segment LARM_LINK4

                                  DEF LARM_WRIST_Y Joint {
                                    jointType "rotate"
                              jointAxis "Z"
                                    jointId  22
                                    translation       0 0 -0.1515875
                                    children [
                                    DEF sensor10 ForceSensor {
sensorId 10 }
                                      DEF LARM_LINK5 Segment {
                                        centerOfMass  0 0 0.1
                                        mass    0.4
                                        momentsOfInertia [ 1 0 0 0 1
0 0 0 1 ]
                                        children DEF ARM_SHAPE5
Transform {
                                          translation 0 0 0.05
                                          rotation 1 0 0 1.5708
                                          children Shape {
                                            appearance Appearance {
                                              material Material {
                                              }
                                            }
                                            geometry Cylinder {
radius 0.025 height 0.05 }
                                          }
                                        }
                                      }
```

```
                                        DEF LARM_WRIST_R Joint {
                                          jointType "rotate"
                                    jointAxis "X"
                                          jointId  23
                                          children [
                                          DEF sensor11 ForceSensor
{ sensorId 11 }
                                            DEF LARM_LINK7 Segment
{
                                              centerOfMass  0 0 -
0.1
                                              mass    0.4
                                              momentsOfInertia [ 1
0 0 0 1 0 0 0 1 ]
                                              children DEF
ARM_SHAPE7 Transform {
                                                children [
                                                  Transform {
                                                    rotation 0 0 1
1.5708
                                                    children Shape
{
                                                      appearance
DEF ARM_APP7 Appearance {
                                                        material
Material {
diffuseColor 0.7 0.9 0.7
                                                    }
                                                  }
                                                  geometry
Cylinder { radius 0.025 height 0.05 }
                                                }
                                              }
                                              Transform {
                                                translation 0
0 -0.08829375
                                                children
Shape {
                                                  appearance
USE ARM_APP7
                                                  geometry Box
{ size 0.05 0.05 0.1265875 }
                                                }
                                              }# Transform
                                            ]
                                          }
                                        } # Segment LARM_LINK7

                                          ]
                                        } # Joint LARM_WRIST_R

                                    ]
                                  } # Joint LARM_WRIST_Y
                                ]
```

```
                    } # Joint LARM_ELBOW
                  ]
                } # Joint LARM_SHOULDER_Y
              ]
            } # Joint LARM_SHOULDER_R
          ]
        } # Joint LARM_SHOULDER_P

      #==================== Right Arm ====================

        DEF RARM_SHOULDER_P Joint {
          jointType "rotate"
      jointAxis "Y"
          jointId 6
          translation   0 -0.215 -0.005
          children [
          DEF sensor12 ForceSensor { sensorId 12 }
            DEF RARM_LINK1 Segment {
              centerOfMass          0.1 0 0
              mass                  3.0
              momentsOfInertia      [ 1 0 0 0 1 0 0 0 1 ]
              children Transform {
                translation 0 0.035 0
                children USE ARM_SHAPE1
              }
            }

            DEF RARM_SHOULDER_R Joint {
              jointType "rotate"
        jointAxis "X"
              jointId 7
              children [
              DEF sensor13 ForceSensor { sensorId 13 }
                DEF RARM_LINK2 Segment {
                  centerOfMass          0 0 -0.1
                  mass                  0.6
                  momentsOfInertia      [ 1 0 0 0 1 0 0 0
1 ]
                  children [
                    USE ARM_SHAPE2
                  ]
                } # Segment RARM_LINK2

                DEF RARM_SHOULDER_Y Joint {
                  jointType "rotate"
                  jointId 8
                  translation     0 0 -0.271825
                  children [
                  DEF sensor14 ForceSensor { sensorId 14 }
                    DEF RARM_LINK3 Segment {
                      centerOfMass          0 0 0
                      mass                  1.0
                      momentsOfInertia      [ 1 0 0 0 1 0
0 0 1 ]
                      children USE ARM_SHAPE3
                    }

                    DEF RARM_ELBOW Joint {
                      jointType "rotate"
                jointAxis "Y"
                        jointId 9
```

```
                                children [
                                DEF sensor15 ForceSensor { sensorId
15 }
                                  DEF RARM_LINK4 Segment {
                                    centerOfMass          0 0 -0.3
                                    mass                  0.6
                                    momentsOfInertia      [ 1 0 0 0
1 0 0 0 1 ]
                                    children USE ARM_SHAPE4
                                  } # Segment RARM_LINK4

                                  DEF RARM_WRIST_Y Joint {
                                    jointType "rotate"
                              jointAxis "Z"
                                    jointId 10
                                    translation  0 0 -0.1515875
                                    children [
                                    DEF sensor16 ForceSensor {
sensorId 16 }
                                      DEF RARM_LINK5 Segment {
                                        centerOfMass          0 0
0.1
                                        mass                  0.4
                                        momentsOfInertia      [ 1 0
0 0 1 0 0 0 1 ]
                                        children USE ARM_SHAPE5
                                      }


                                          DEF RARM_WRIST_R Joint {
                                            jointType "rotate"
                                      jointAxis "X"
                                            jointId 11
                                            children [
                                            DEF sensor17 ForceSensor
{ sensorId 17 }
                                              DEF RARM_LINK7 Segment
{
                                                centerOfMass
0 0 -0.1
                                                mass
0.4
                                                momentsOfInertia
[ 1 0 0 0 1 0 0 0 1 ]
                                                children [
                                                  USE ARM_SHAPE7
                                                ]
                                              } # Segment RARM_LINK7
                                            ]
                                          } # Joint RARM_WRIST_R

                                    ]
                                  } # Joint RARM_WRIST_Y
                                ]
                              } # Joint RARM_ELBOW
                            ]
                          } # Joint RARM_SHOULDER_Y
                        ]
                      } # Joint RARM_SHOULDER_R
                    ]
                  } # Joint RARM_SHOULDER_P
```

```
            ]
            }# Joint CHEST_P
          ]
        } # Joint CHEST_Y
      ]
    } # Joint WAIST_Y
  ]
} # Joint WAIST_P

#==================== Left Leg ====================

DEF LLEG_HIP_R Joint {
  jointType "rotate"
jointAxis "X"
  jointId 12
  translation   0 0.10877 -0.01
  children [
  DEF sensor18 ForceSensor { sensorId 18 }

    DEF LLEG_LINK1 Segment {
      centerOfMass          0 0.1 0
      mass                  2.5
      momentsOfInertia      [ 1 0 0 0 1 0 0 0 1 ]
      children DEF LEG_SHAPE1 Transform {
        rotation 0 0 1 1.5708
        children Shape {
          appearance Appearance {
            material Material {
            }
          }
          geometry Cylinder { radius 0.05 height 0.1 }
        }
      }
    }

    DEF LLEG_HIP_P Joint {
      jointType "rotate"
    jointAxis "Y"
      jointId 13
      children    [
          DEF sensor19 ForceSensor { sensorId 19 }


           DEF LLEG_LINK2 Segment {
             centerOfMass          0 0 -0.15
             mass                  2.0
             momentsOfInertia      [ 1 0 0 0 1 0 0 0 1 ]
             children DEF LEG_SHAPE2 Shape {
               appearance Appearance {
                 material Material {
                 }
               }
               geometry Cylinder { radius 0.05 height 0.1 }
             }
           }

           DEF LLEG_HIP_Y Joint {
             jointType "rotate"
             jointId 14
             translation     0 0 -0.359875
             children [
```

```
                        DEF sensor20 ForceSensor { sensorId 20 }
                    DEF LLEG_LINK3 Segment {
                      centerOfMass          0 0.04 0
                      mass                  5.1
                      momentsOfInertia      [ 1 0 0 0 1 0 0 0 1 ]
                      children DEF LEG_SHAPE3 Transform {
                        children [
                          Transform {
                            translation 0 0 0.1
                            rotation 1 0 0 1.5708
                            children Shape {
                              appearance DEF LEG_APP3 Appearance {
                                material  Material {
                                  diffuseColor   0.8 0.9 0.8
                                }
                              }
                              geometry Cylinder { radius 0.05
height 0.1 }
                            }
                          }
                          Transform {
                            translation 0 0 0.2299375
                            children      Shape {
                              appearance USE LEG_APP3
                              geometry Box { size 0.2 0.1 0.159875
}
                            }
                          }
                        ]
                      }
                    } # Segment LLEG_LINK3

                    DEF LLEG_KNEE Joint {
                      jointType "rotate"
                jointAxis "Y"
                      jointId 15
                      children [
                     DEF sensor21 ForceSensor { sensorId 21 }
                        DEF LLEG_LINK4 Segment {
                          centerOfMass          0 0 -0.3
                          mass                  7.0
                          momentsOfInertia      [ 1 0 0 0 1 0 0 0 1
]
                          children DEF LEG_SHAPE4 Transform {
                            children [
                              Shape {
                                appearance DEF LEG_APP4 Appearance
{
                                  material    Material {
                                    diffuseColor     0.8 1.0 0.8
                                  }
                                }
                                geometry Cylinder { radius 0.05
height 0.1 }
                              }
                              Transform {
                                translation      0 0 -0.175
                                children        Shape {
                                  appearance USE LEG_APP4
                                  geometry Box { size 0.2 0.1 0.25
}
```

```
                                }
                              }
                            ]
                          }
                        } # Segment LLEG_LINK4

                        DEF LLEG_ANKLE_P Joint {
                          jointType "rotate"
                    jointAxis "Y"
                          jointId 16
                          translation       0 0 -0.35
                          children [
                            DEF sensor22 ForceSensor { sensorId 22 }
                            DEF LLEG_LINK5 Segment {
                              centerOfMass          -0.15 0 0
                              mass                  2.5
                              momentsOfInertia      [ 1 0 0 0 1 0 0
0 1 ]
                              children DEF LEG_SHAPE5 Shape {
                                appearance Appearance {
                                  material Material {
                                  }
                                }
                                geometry Cylinder { radius 0.05
height 0.1 }
                              }
                            }
                            DEF LLEG_ANKLE_R Joint {
                              jointType "rotate"
                      jointAxis "X"
                              jointId 17
                              children [
                             DEF sensor23 ForceSensor { sensorId 23 }
                                 DEF LLEG_LINK6 Segment {
                                   centerOfMass          0.28 0 -0.2
                                   mass                  1.9
                                   momentsOfInertia      [ 1 0 0 0 1
0 0 0 1 ]
                                   children DEF LEG_SHAPE6 Transform
{
                                     children [
                                       Transform {
                                  #translation 0 0 0
                                            rotation 0 0 1 1.5708
                                            children Shape {
                                              appearance DEF LEG_APP6
Appearance {
                                                material      Material
{
                                                  diffuseColor 0.0 0.5
0.0
                                              }
                                            }
                                            geometry Cylinder {
radius 0.05 height 0.1 }
                                          }
                                        }
                                         Transform {
                                          translation       0.12 0 0
                                          children  Shape {
                                            appearance USE LEG_APP6
```

```
                                        geometry Box { size 0.14
0.11 0.1 }
                                      } # Shape
                                    }#Transform

                                    Transform {
                                      translation       0.055 0 -
0.0549375
                                      children  Shape {
                                        appearance USE LEG_APP6
                                        geometry Box { size 0.3
0.14 0.009875 }
                                      } # Shape
                                    }#Transform
                                  ]
                                }
                              } # Segment LLEG_LINK6

                            ]
                          } # Joint LLEG_ANKLE_R

                        ]
                      } # Joint LLEG_ANKLE_P

                    ]
                  } # Joint LLEG_KNEE

                ]
              } # Joint LLEG_HIP_Y

            ]
          } # Joint LLEG_HIP_P
        ]
      } # Joint LLEG_HIP_R

      #==================== Right Leg ====================

      DEF RLEG_HIP_R Joint {
        jointType "rotate"
      jointAxis "X"
        jointId 0
        translation   0 -0.10877 -0.01
        children [
          DEF sensor24 ForceSensor { sensorId 24 }
          DEF RLEG_LINK1 Segment {
            centerOfMass          0 -0.1 0
            mass                  2.5
            momentsOfInertia      [ 1 0 0 0 1 0 0 0 1 ]
            children USE LEG_SHAPE1
          }

          DEF RLEG_HIP_P Joint {
            jointType "rotate"
          jointAxis "Y"
            jointId 1
            children [
                DEF sensor25 ForceSensor { sensorId 25 }
              DEF RLEG_LINK2 Segment {
                centerOfMass          0 0 -0.15
                mass                  2.0
                momentsOfInertia      [ 1 0 0 0 1 0 0 0 1 ]
```

```
                children USE LEG_SHAPE2
              }
              DEF RLEG_HIP_Y Joint {
                jointType "rotate"
                jointId 2
                translation     0 0 -0.359875
                children [
                DEF sensor26 ForceSensor { sensorId 26 }
                  DEF RLEG_LINK3 Segment {
                    centerOfMass          0 -0.04 0
                    mass                  5.1
                    momentsOfInertia      [ 1 0 0 0 1 0 0 0 1 ]
                    children [
                      USE LEG_SHAPE3
                    ]
                  }

                  DEF RLEG_KNEE Joint {
                    jointType "rotate"
                jointAxis "Y"
                    jointId 3
                    children [
            DEF sensor27 ForceSensor { sensorId 27 }
                      DEF RLEG_LINK4 Segment {
                        centerOfMass          0 0 -0.3
                        mass                  7.0
                        momentsOfInertia      [ 1 0 0 0 1 0 0 0 1 ]
                        children [
                          USE LEG_SHAPE4
                        ]
                      } # Segment RLEG_LINK4

                      DEF RLEG_ANKLE_P Joint {
                        jointType "rotate"
                    jointAxis "Y"
                        jointId 4
                        translation     0 0 -0.35
                        children [
          DEF sensor28 ForceSensor { sensorId 28 }
                          DEF RLEG_LINK5 Segment {
                            centerOfMass          -0.15 0 0
                            mass                  2.5
                            momentsOfInertia      [ 1 0 0 0 1 0 0 0
1 ]
                            children USE LEG_SHAPE5
                          }

                          DEF RLEG_ANKLE_R Joint {
                            jointType "rotate"
                          jointAxis "X"
                            jointId 5
                            children [
                DEF sensor29 ForceSensor { sensorId 29 }
                              DEF RLEG_LINK6 Segment {
                                centerOfMass          0.28 0 -0.2
                                mass                  1.9
                                momentsOfInertia      [ 1 0 0 0 1 0
0 0 1 ]
                                children [
                                  USE LEG_SHAPE6
                                ]
```

```
                                } # Segment RLEG_LINK6

                              ]
                            } # Joint RLEG_ANKLE_R

                          ]
                        } # Joint RLEG_ANKLE_P

                      ]
                    } # Joint RLEG_KNEE

                  ]
                } # Joint RLEG_HIP_Y

              ]
            } # Joint RLEG_HIP_P

          ]
        } # Joint RLEG_HIP_R
      ]
    } # Joint WAIST
  ]


  # List up all the joints' name you use
  joints [
    USE WAIST,
    USE WAIST_P,
    USE WAIST_Y,
    USE CHEST_Y,
    USE CHEST_P,

    USE LARM_SHOULDER_P,
    USE LARM_SHOULDER_R,
    USE LARM_SHOULDER_Y,
    USE LARM_ELBOW,
    USE LARM_WRIST_Y,
    USE LARM_WRIST_R,

    USE RARM_SHOULDER_P,
    USE RARM_SHOULDER_R,
    USE RARM_SHOULDER_Y,
    USE RARM_ELBOW,
    USE RARM_WRIST_Y,
    USE RARM_WRIST_R,

    USE LLEG_HIP_R,
    USE LLEG_HIP_P,
    USE LLEG_HIP_Y,
    USE LLEG_KNEE,
    USE LLEG_ANKLE_P,
    USE LLEG_ANKLE_R,

    USE RLEG_HIP_R,
    USE RLEG_HIP_P,
    USE RLEG_HIP_Y,
    USE RLEG_KNEE,
    USE RLEG_ANKLE_P,
    USE RLEG_ANKLE_R
  ]
```

```
  # List up all the segments' name you use
  segments [
    USE WAIST_LINK0,
    USE WAIST_LINK1,
    USE WAIST_LINK2,
    USE WAIST_LINK3,
    USE WAIST_LINK4,

    USE LARM_LINK1,
    USE LARM_LINK2,
    USE LARM_LINK3,
    USE LARM_LINK4,
    USE LARM_LINK5,
    USE LARM_LINK7,

    USE RARM_LINK1,
    USE RARM_LINK2,
    USE RARM_LINK3,
    USE RARM_LINK4,
    USE RARM_LINK5,
    USE RARM_LINK7,

    USE LLEG_LINK1,
    USE LLEG_LINK2,
    USE LLEG_LINK3,
    USE LLEG_LINK4,
    USE LLEG_LINK5,
    USE LLEG_LINK6,

    USE RLEG_LINK1,
    USE RLEG_LINK2,
    USE RLEG_LINK3,
    USE RLEG_LINK4,
    USE RLEG_LINK5,
    USE RLEG_LINK6
  ]
}
```

9.2 Código del archivo .xml encargado de abrir el proyecto usado

```
<?xml version="1.0" encoding="UTF-8"?>
<grxui>
    <mode name="Simulation">
        <item class="com.generalrobotix.ui.item.GrxWorldStateItem"
name="untitled" select="true">
            <property name="logTimeStep" value="0.005 "/>
            <property name="integrate" value="true"/>
            <property name="viewsimulate" value="false"/>
            <property name="totalTime" value="7.0 "/>
            <property name="timeStep" value="0.005 "/>
            <property name="method" value="RUNGE_KUTTA"/>
            <property name="gravity" value="9.8 "/>
            <property name="viewsimulationTimeStep" value="0.033 "/>
        </item>
        <item class="com.generalrobotix.ui.item.GrxModelItem"
name="floor" select="true" url="$(OPENHRPHOME)/etc/floor.wrl">
            <property name="isRobot" value="false"/>
            <property name="WAIST.rotation" value="0.0 1.0 0.0 0.0 "/>
            <property name="WAIST.translation" value="0.0 0.0 -0.1 "/>
        </item>
        <item class="com.generalrobotix.ui.item.GrxModelItem"
name="sample" select="true" url="$(OPENHRPHOME)/etc/sample.wrl">
            <property name="isRobot" value="true"/>
            <property name="controller" value="SampleHGController"/>
            <property name="controlTime" value="0.002"/>
            <property name="setupDirectory"
value="$(OPENHRPHOME)/Controller/rtc/SampleHG"/>
            <property name="setupCommand" value="SampleHG$(BIN_SFX)"/>
            <property name="RLEG_HIP_R.angle" value="0.0 "/>
            <property name="RARM_SHOULDER_R.mode" value="HighGain"/>
            <property name="LLEG_KNEE.mode" value="HighGain"/>
            <property name="RARM_ELBOW.angle" value="-1.5708 "/>
            <property name="LLEG_ANKLE_P.mode" value="HighGain"/>
            <property name="RLEG_ANKLE_P.angle" value="-0.0424675 "/>
            <property name="LLEG_ANKLE_R.mode" value="HighGain"/>
            <property name="RLEG_ANKLE_R.angle" value="0.0 "/>
            <property name="LLEG_HIP_Y.mode" value="HighGain"/>
            <property name="RLEG_HIP_P.mode" value="HighGain"/>
            <property name="RARM_WRIST_P.angle" value="0.0 "/>
            <property name="CHEST.mode" value="HighGain"/>
            <property name="RARM_WRIST_R.angle" value="0.0 "/>
            <property name="RARM_WRIST_Y.angle" value="0.0 "/>
            <property name="RLEG_KNEE.angle" value="0.0785047 "/>
            <property name="RLEG_HIP_R.mode" value="HighGain"/>
            <property name="LARM_SHOULDER_P.angle" value="0.174533 "/>
            <property name="LARM_SHOULDER_R.angle" value="-0.00349066
"/>
            <property name="LARM_WRIST_P.mode" value="HighGain"/>
            <property name="LARM_SHOULDER_Y.angle" value="0.0 "/>
            <property name="LLEG_HIP_P.angle" value="-0.0360373 "/>
            <property name="LARM_WRIST_R.mode" value="HighGain"/>
            <property name="LLEG_HIP_R.angle" value="0.0 "/>
            <property name="WAIST.rotation" value="0.0 1.0 0.0 0.0 "/>
```

```
            <property name="LLEG_HIP_Y.angle" value="0.0 "/>
            <property name="LARM_ELBOW.angle" value="-1.5708 "/>
            <property name="LARM_SHOULDER_Y.mode" value="HighGain"/>
            <property name="RLEG_KNEE.mode" value="HighGain"/>
            <property name="CHEST.angle" value="0.0 "/>
            <property name="WAIST_P.mode" value="HighGain"/>
            <property name="LLEG_HIP_P.mode" value="HighGain"/>
            <property name="LLEG_ANKLE_P.angle" value="-0.0424675 "/>
            <property name="RARM_SHOULDER_P.angle" value="0.174533 "/>
            <property name="LLEG_ANKLE_R.angle" value="0.0 "/>
            <property name="RARM_SHOULDER_R.angle" value="-0.00349066
"/>
            <property name="WAIST_R.mode" value="HighGain"/>
            <property name="LLEG_KNEE.angle" value="0.0785047 "/>
            <property name="LLEG_HIP_R.mode" value="HighGain"/>
            <property name="RARM_SHOULDER_Y.angle" value="0.0 "/>
            <property name="LARM_WRIST_P.angle" value="0.0 "/>
            <property name="LARM_WRIST_R.angle" value="0.0 "/>
            <property name="LARM_WRIST_Y.angle" value="0.0 "/>
            <property name="RARM_WRIST_Y.mode" value="HighGain"/>
            <property name="RARM_ELBOW.mode" value="HighGain"/>
            <property name="RARM_SHOULDER_Y.mode" value="HighGain"/>
            <property name="WAIST.mode" value="HighGain"/>
            <property name="WAIST_P.angle" value="0.0 "/>
            <property name="LARM_SHOULDER_P.mode" value="HighGain"/>
            <property name="WAIST_R.angle" value="0.0 "/>
            <property name="LARM_SHOULDER_R.mode" value="HighGain"/>
            <property name="WAIST.translation" value="0.0 0.0 0.77975
"/>
            <property name="RLEG_HIP_Y.mode" value="HighGain"/>
            <property name="RLEG_ANKLE_P.mode" value="HighGain"/>
            <property name="LARM_WRIST_Y.mode" value="HighGain"/>
            <property name="RLEG_ANKLE_R.mode" value="HighGain"/>
            <property name="RARM_WRIST_P.mode" value="HighGain"/>
            <property name="LARM_ELBOW.mode" value="HighGain"/>
            <property name="RARM_WRIST_R.mode" value="HighGain"/>
            <property name="RARM_SHOULDER_P.mode" value="HighGain"/>
            <property name="RLEG_HIP_P.angle" value="-0.0360373 "/>
            <property name="RLEG_HIP_Y.angle" value="0.0 "/>
        </item>
        <item class="com.generalrobotix.ui.item.GrxCollisionPairItem"
name="CP#floor#sample" select="true">
            <property name="springConstant" value="0 0 0 0 0 0"/>
            <property name="slidingFriction" value="0.5"/>
            <property name="jointName2" value=""/>
            <property name="jointName1" value=""/>
            <property name="sprintDamperModel" value="false"/>
            <property name="damperConstant" value="0 0 0 0 0 0"/>
            <property name="objectName2" value="sample"/>
            <property name="objectName1" value="floor"/>
            <property name="staticFriction" value="0.5"/>
        </item>
        <item class="com.generalrobotix.ui.item.GrxGraphItem"
name="GraphList1" select="true">
            <property name="Graph0.dataItems" value=""/>
            <property name="Graph1.dataItems" value=""/>
            <property name="Graph2.dataItems" value=""/>
            <property name="Graph3.dataItems" value=""/>
        </item>
    </mode>
</grxui>
```

9.3 Código SampleHG.cpp del controlador con los datos actualizados

```
// -*- mode: c++; indent-tabs-mode: t; tab-width: 4; c-basic-offset:
4; -*-
/*
 * Copyright (c) 2008, AIST, the University of Tokyo and General
Robotix Inc.
 * All rights reserved. This program is made available under the terms
of the
 * Eclipse Public License v1.0 which accompanies this distribution,
and is
 * available at http://www.eclipse.org/legal/epl-v10.html
 * Contributors:
 * National Institute of Advanced Industrial Science and Technology
(AIST)
 * General Robotix Inc.
 */
/*!
 * @file  SampleHG.cpp
 * @brief Sample LF component
 * $Date$
 *
 * $Id$
 */

#include "SampleHG.h"

#include <iostream>

#define DOF (28)

#define ANGLE_FILE "etc/angle.dat"
#define VEL_FILE   "etc/vel.dat"
#define ACC_FILE   "etc/acc.dat"

namespace {
  const bool CONTROLLER_BRIDGE_DEBUG = false;
}


// Module specification
// <rtc-template block="module_spec">
static const char* samplepd_spec[] =
  {
    "implementation_id", "SampleHG",
    "type_name",         "SampleHG",
    "description",       "Sample HG component",
    "version",           "0.1",
    "vendor",            "AIST",
    "category",          "Generic",
    "activity_type",     "DataFlowComponent",
    "max_instance",      "10",
    "language",          "C++",
    "lang_type",         "compile",
    // Configuration variables
```

```
    ""
  };
// </rtc-template>

SampleHG::SampleHG(RTC::Manager* manager)
  : RTC::DataFlowComponentBase(manager),
    // <rtc-template block="initializer">
    m_angleOut("angle", m_angle),
    m_velOut("vel", m_vel),
    m_accOut("acc", m_acc)

    // </rtc-template>
{
  if( CONTROLLER_BRIDGE_DEBUG )
  {
    std::cout << "SampleHG::SampleHG" << std::endl;
  }
  // Registration: InPort/OutPort/Service
  // <rtc-template block="registration">
  // Set InPort buffers

  // Set OutPort buffer
  registerOutPort("angle", m_angleOut);
  registerOutPort("vel", m_velOut);
  registerOutPort("acc", m_accOut);
  // Set service provider to Ports

  // Set service consumers to Ports

  // Set CORBA Service Ports

  // </rtc-template>

  if (access(ANGLE_FILE, 0)){
    std::cerr << ANGLE_FILE << " not found" << std::endl;
  }else{
    angle.open(ANGLE_FILE);
  }

  if (access(VEL_FILE, 0)){
    std::cerr << VEL_FILE << " not found" << std::endl;
  }else{
    vel.open(VEL_FILE);
  }

  if (access(ACC_FILE, 0))
  {
    std::cerr << ACC_FILE << " not found" << std::endl;
  }else{
    acc.open(ACC_FILE);
  }

  m_angle.data.length(DOF);
  m_vel.data.length(DOF);
  m_acc.data.length(DOF);

}

SampleHG::~SampleHG()
{
```

```
      if (angle.is_open())  angle.close();
      if (vel.is_open())    vel.close();
      if (acc.is_open())    acc.close();
}


RTC::ReturnCode_t SampleHG::onInitialize()
{
  // <rtc-template block="bind_config">
  // Bind variables and configuration variable
  if( CONTROLLER_BRIDGE_DEBUG )
  {
    std::cout << "onInitialize" << std::endl;
  }

  // </rtc-template>
  return RTC::RTC_OK;
}



/*
RTC::ReturnCode_t SampleHG::onFinalize()
{
  return RTC::RTC_OK;
}
*/

/*
RTC::ReturnCode_t SampleHG::onStartup(RTC::UniqueId ec_id)
{
  return RTC::RTC_OK;
}
*/

/*
RTC::ReturnCode_t SampleHG::onShutdown(RTC::UniqueId ec_id)
{
  return RTC::RTC_OK;
}
*/

RTC::ReturnCode_t SampleHG::onActivated(RTC::UniqueId ec_id)
{

      std::cout << "on Activated" << std::endl;
      angle.seekg(0);
      vel.seekg(0);
      acc.seekg(0);

      return RTC::RTC_OK;
}

/*
RTC::ReturnCode_t SampleHG::onDeactivated(RTC::UniqueId ec_id)
{
  return RTC::RTC_OK;
}
*/
```

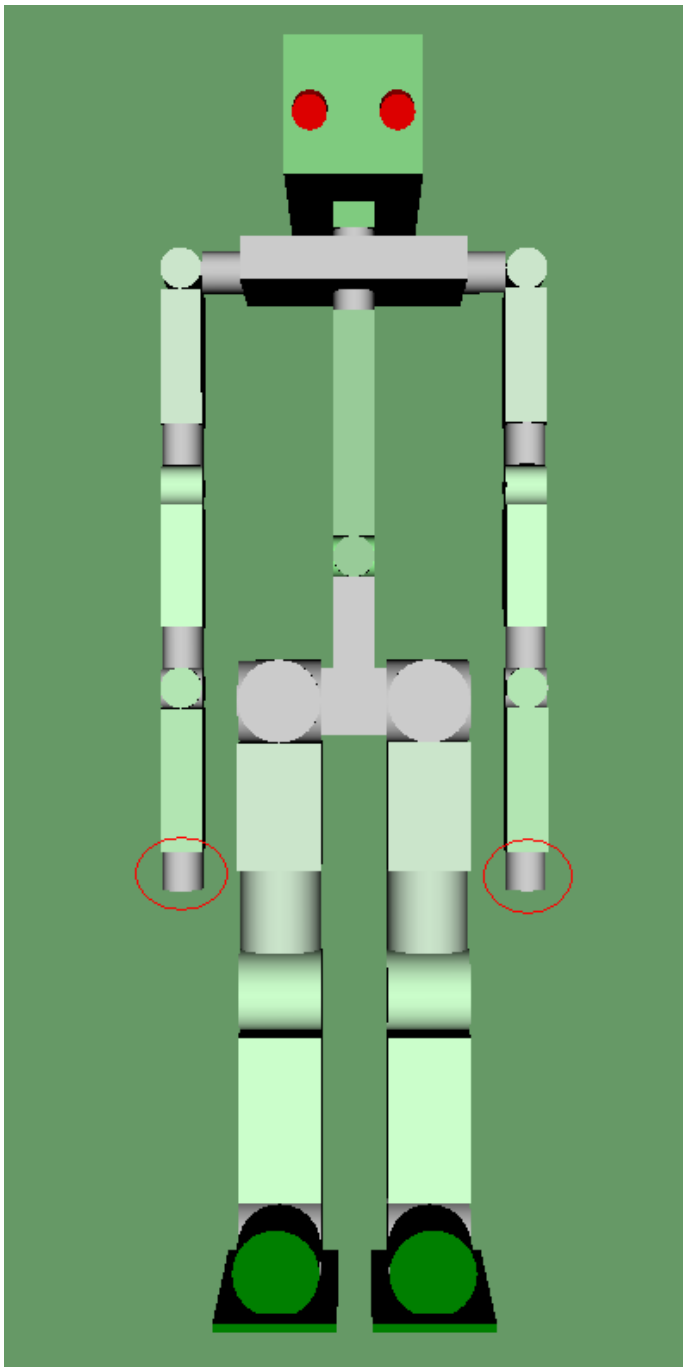
```
RTC::ReturnCode_t SampleHG::onExecute(RTC::UniqueId ec_id)
{
  if( CONTROLLER_BRIDGE_DEBUG )
  {
    std::cout << "SampleHG::onExecute" << std::endl;
  }
  //
,±,ÌŠÖ ”,Ì U,é•‘,¢,ÍController_impl::control,Ì”h ¶ æ‰¼‘zŠÖ ”,É‘Î‰ž,·,é
  double dummy;
  angle >> dummy; vel >> dummy; acc >> dummy; // skip time
  int i;

  //Šeƒtƒ@ƒCƒ‹,©,çƒf [ƒ^,ðˆê s“Ç,Ý ž,ñ,Åƒ| [ƒg,É—¬,·
  for (i=0; i<DOF; i++)
  {
      angle >> m_angle.data[i];
      vel   >> m_vel.data[i];
      acc   >> m_acc.data[i];
  }

  m_angleOut.write();
  m_velOut.write();
  m_accOut.write();

  return RTC::RTC_OK;
}


/*
  RTC::ReturnCode_t SampleHG::onAborting(RTC::UniqueId ec_id)
  {
  return RTC::RTC_OK;
  }
*/

/*
  RTC::ReturnCode_t SampleHG::onError(RTC::UniqueId ec_id)
  {
  return RTC::RTC_OK;
  }
*/

/*
  RTC::ReturnCode_t SampleHG::onReset(RTC::UniqueId ec_id)
  {
  return RTC::RTC_OK;
  }
*/

/*
  RTC::ReturnCode_t SampleHG::onStateUpdate(RTC::UniqueId ec_id)
  {
  return RTC::RTC_OK;
  }
*/

/*
  RTC::ReturnCode_t SampleHG::onRateChanged(RTC::UniqueId ec_id)
  {
  return RTC::RTC_OK;
  }
```

```
*/


extern "C"
{

      DllExport void SampleHGInit(RTC::Manager* manager)
      {
            RTC::Properties profile(samplepd_spec);
            manager->registerFactory(profile,
                                                    RTC::Create<SampleHG>,

RTC::Delete<SampleHG>);
      }

};
```

9.4 Manual de adición de eslabones a un modelo .vrml en OpenHRP3

A continuación se muestra los pasos a seguir si se quieren añadir eslabones a un modelo en vrml. Para la explicación se emplea el modelo que viene por defecto en el simulador OpenHRP3. Se ha incorporado un nuevo eslabón en el extremo de cada uno de los brazos. Estas nuevas piezas son un cilindro al cual se le ha asignado un grado de libertad.

El modelo resultante después de la modificación es el que se muestra a continuación.



Al modificar el modelo y añadirle dos nuevas piezas con un grado de libertad cada una, se han tenido que crear dos nuevos **Joints** con sus respectivos **Segment**, los cuales se han insertado siguiendo la estructura de programación.

A continuación se muestra como resulta la estructura después de añadirle los dos nuevos joints.

```
| # Root
+-humanoidBody
  |
  | # Upper half body
  +-Joint WAIST : Segment WAIST_LINK0
  |   Joint WAIST_P : Segment WAIST_LINK1
  |     Joint WAIST_R : Segment WAIST_LINK2
  |       Joint CHEST : Segment WAIST_LINK3
  |       |
  |       | # Cameras
  |       +-VisionSensor VISION_SENSOR1
  |       +-VisionSensor VISION_SENSOR2
  |       |
  |       | # Left arm
  |       +-Joint LARM_SHOULDER_P : Segment LARM_LINK1
  |       |   Joint LARM_SHOULDER_R : Segment LARM_LINK2
  |       |     Joint LARM_SHOULDER_Y : Segment LARM_LINK3
  |       |       Joint LARM_ELBOW : Segment LARM_LINK4
  |       |         Joint LARM_WRIST_Y : Segment LARM_LINK5
  |       |           Joint LARM_WRIST_P : Segment LARM_LINK6
  |       |             Joint LARM_WRIST_R : Segment LARM_LINK7
  |       |               Joint LBRAZO: Segment LARM_LINK8
  |       |
  |       | # Right arm
  |       +-Joint RARM_SHOULDER_P : Segment RARM_LINK1
  |           Joint RARM_SHOULDER_R : Segment RARM_LINK2
  |             Joint RARM_SHOULDER_Y : Segment RARM_LINK3
  |               Joint RARM_ELBOW : Segment RARM_LINK4
  |                 Joint RARM_WRIST_Y : Segment RARM_LINK5
  |                   Joint RARM_WRIST_P : Segment RARM_LINK6
  |                     Joint RARM_WRIST_R : Segment RARM_LINK7
  |                       Joint DBRAZO: Segment RARM_LINK8
  |
  | # Left Leg
  +-Joint LLEG_HIP_R : Segment LLEG_LINK1
```

```
|   Joint LLEG_HIP_P : Segment LLEG_LINK2
|     Joint LLEG_HIP_Y : Segment LLEG_LINK3
|       Joint LLEG_KNEE : Segment LLEG_LINK4
|         Joint LLEG_ANKLE_P : Segment LLEG_LINK5
|           Joint LLEG_ANKLE_R : Segment LLEG_LINK6
|
| # Right Leg
+-Joint RLEG_HIP_R : Segment RLEG_LINK1
      Joint RLEG_HIP_P : Segment RLEG_LINK2
        Joint RLEG_HIP_Y : Segment RLEG_LINK3
          Joint RLEG_KNEE : Segment RLEG_LINK4
            Joint RLEG_ANKLE_P : Segment RLEG_LINK5
              Joint RLEG_ANKLE_R : Segment RLEG_LINK6
```

En primer lugar hay que crear la pieza del brazo izquierdo. El código utilizado para crear esta nueva pieza es el siguiente.

```
DEF LBRAZO Joint {
      jointType "rotate"
      jointAxis "Z"
      jointId  29

      translation       0 0 -0.247
      children [
      DEF LARM_LINK8 Segment {
          centerOfMass  0 0 0.1
          mass    0.0
          momentsOfInertia [ 1 0 0 0 1 0 0 0 1 ]

          children DEF ARM_SHAPE8
          Transform {
               translation 0 0 0.05
               rotation 1 0 0 1.5708
               children Shape {
               appearance Appearance {
               material Material {
                     }
                   }
               geometry Cylinder { radius 0.025 height 0.1 }
                 }
            }
         }
```

```
        ]
    } # Joint LBRAZO
```

A continuación se pasa a crear la nueva para el brazo derecho. El código utilizado es distinto ya que al crear la pieza del brazo izquierdo se crea una determinada apariencia para la pieza, que se utilizará en la pieza del brazo derecho para que las dos tengan la misma forma.

```
DEF DBRAZO Joint {
      jointType "rotate"
      jointAxis "Z"
      jointId  30

      translation       0 0 -0.247
      children [
      DEF RARM_LINK8 Segment {
          centerOfMass  0 0 0.1
          mass    0.0
          momentsOfInertia [ 1 0 0 0 1 0 0 0 1 ]

          children USE ARM_SHAPE8

               }

          ]
  } # Joint DBRAZO
```

Además del código anterior, para introducir una nueva pieza se debe añadir el nombre de los nuevos joints a la lista que aparece al final del código donde se muestra la lista de los joints utilizados en el código:

```
# List up all the joints' name you use
joints [
USE WAIST,
USE WAIST_P,
USE WAIST_R,
USE CHEST,
```

```
USE LARM_SHOULDER_P,
USE LARM_SHOULDER_R,
USE LARM_SHOULDER_Y,
USE LARM_ELBOW,
USE LARM_WRIST_Y,
USE LARM_WRIST_P,
USE LARM_WRIST_R,
USE LBRAZO,

USE RARM_SHOULDER_P,
USE RARM_SHOULDER_R,
USE RARM_SHOULDER_Y,
USE RARM_ELBOW,
USE RARM_WRIST_Y,
USE RARM_WRIST_P,
USE RARM_WRIST_R,
USE DBRAZO,

USE LLEG_HIP_R,
USE LLEG_HIP_P,
USE LLEG_HIP_Y,
USE LLEG_KNEE,
USE LLEG_ANKLE_P,
USE LLEG_ANKLE_R,

USE RLEG_HIP_R,
USE RLEG_HIP_P,
USE RLEG_HIP_Y,
USE RLEG_KNEE,
USE RLEG_ANKLE_P,
USE RLEG_ANKLE_R
]
```

De igual manera se modifica la lista de los segmentos utilizados en el código:

```
  # List up all the segments' name you use
  segments [
    USE WAIST_LINK0,
    USE WAIST_LINK1,
    USE WAIST_LINK2,
```

```
  USE WAIST_LINK3,

  USE LARM_LINK1,
  USE LARM_LINK2,
  USE LARM_LINK3,
  USE LARM_LINK4,
  USE LARM_LINK5,
  USE LARM_LINK6,
  USE LARM_LINK7,
  USE LARM_LINK8,

  USE RARM_LINK1,
  USE RARM_LINK2,
  USE RARM_LINK3,
  USE RARM_LINK4,
  USE RARM_LINK5,
  USE RARM_LINK6,
  USE RARM_LINK7,
  USE RARM_LINK8,

  USE LLEG_LINK1,
  USE LLEG_LINK2,
  USE LLEG_LINK3,
  USE LLEG_LINK4,
  USE LLEG_LINK5,
  USE LLEG_LINK6,

  USE RLEG_LINK1,
  USE RLEG_LINK2,
  USE RLEG_LINK3,
  USE RLEG_LINK4,
  USE RLEG_LINK5,
  USE RLEG_LINK6
]
```